General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)

UNCLASSIFIED

NASA CR-
141787

SPACE VEHICLE VITERBI DECODER

FINAL REPORT

ER75-4176

16 APRIL 1975

PREPARED UNDER

NASA CONTRACT NO. NAS 9-13923

(NASA-CR-141787) SPACE VEHICLE VITERBI DECODER Final Report (Raytheon Co.) 64 p HC $4.25 CSCL 09F

N75-23672

Unclas
G3/19 21774

RAYTHEON

RAYTHEON COMPANY

EQUIPMENT DIVISION

SPACE VEHICLE VITERBI DECODER

FINAL REPORT

ER75-4176

16 APRIL 1975

PREPARED UNDER

NASA CONTRACT NO. NAS 9-13923

FOR

NASA MANNED SPACECRAFT CENTER

HOUSTON, TEXAS

RAYTHEON COMPANY

EQUIPMENT DEVELOPMENT LABORATORY

SUDBURY, MA 01776

I. INTRODUCTION

The primary objective of the Space Vehicle Viterbi Decoder project was to design and fabricate an extremely low-power, constraint-length 7, rate 1/3 Viterbi decoder brassboard capable of operating at information rates of up to 100 kb/s. The brassboard was to be partitioned to facilitate a later transition to an LSI version requiring even less power. Additional objectives included an evaluation of the effect of soft-decision thresholds, path memory lengths and output selection algorithms on the bit error rate and the comparison of a new branch synchronization algorithm with a more conventional approach. The following pages describe in detail the implementation of the decoder and its test set (including novel all-digital noise source) and present the results of the various system tests and evaluations. The report concludes with a summary of the results obtained and with recommendations for the implementation of the LSI version of this decoder.

II. VITERBI DECODER DESIGN AND IMPLEMENTATION

The decoder configuration chosen for the baseline design is a series - parallel implementation of the Viterbi decoding algorithm, tailored to the rate 1/3, constraint length 7, Odenwalder convolutional code. This configuration represents a compromise between a fully parallel decoder with lowest speed/power requirements, and a series decoder with lower hardware requirements.

One of the key elements used to implement the decoder baseline design is a random access memory (RAM). The organization of the RAM units was a determining factor in selection of an 8 x 8 series/parallel configuration.

CMOS integrated circuits were chosen for implementing the decoder design to minimize power requirements. The same logic functions can be used in a later LSI version of the decoder and implemented with either hybrid or monolithic technology.

2.1 General Description*

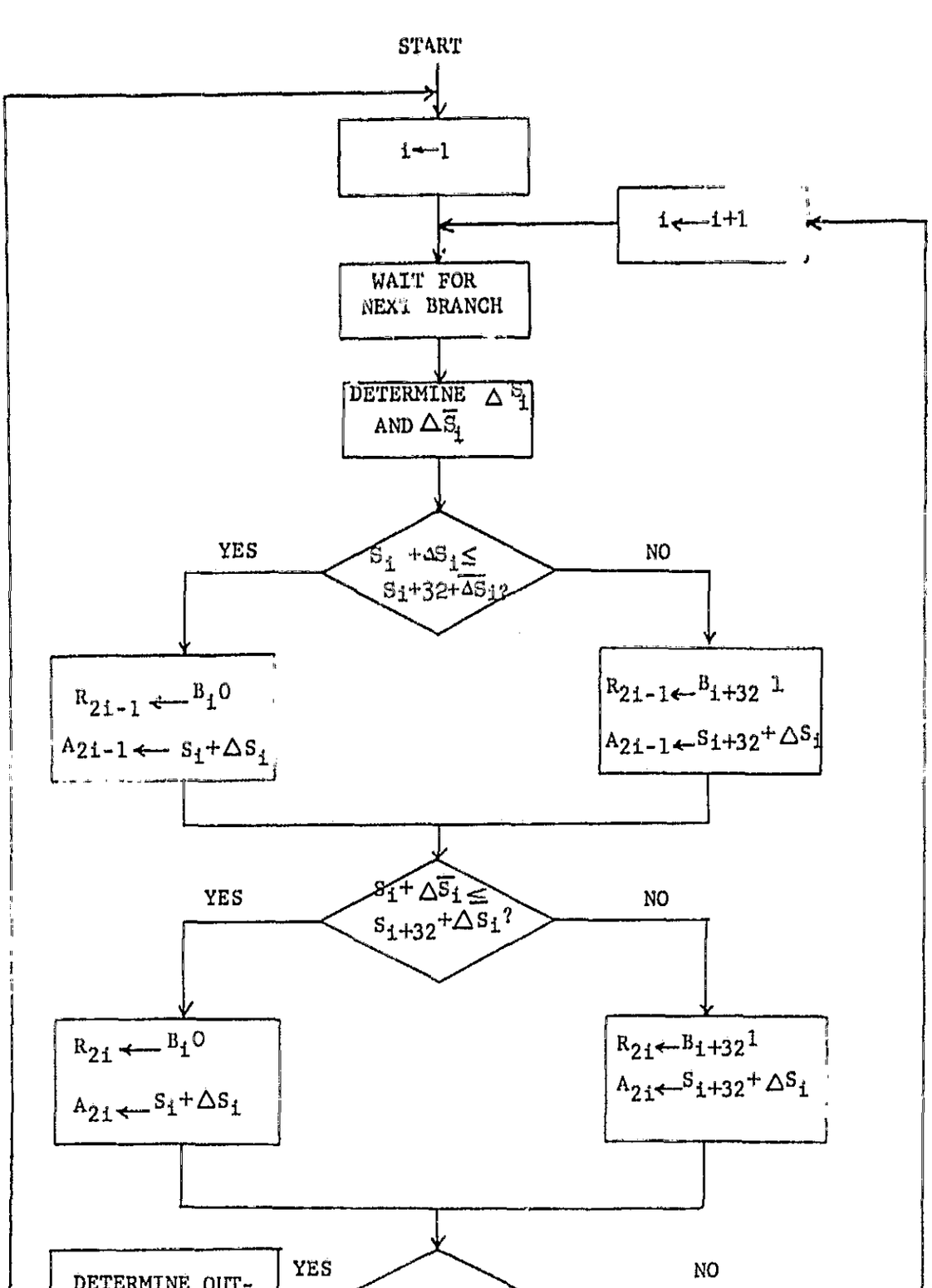
The basic algorithm implemented by the decoder can be explained with the aid of the decoding algorithm flow chart, Figure 2-1, and the simplified decoder block diagram in Figure 2-2.

The branch metric calculator receives the quantized received code bits and generates from the 64 metric increments, Δs.

The input data representation is assumed to be such that the natural binary progression corresponds to the progression from a "strong zero" to a "strong one" (i.e., 000 = strong zero, 001 = moderately strong zero, 010 = moderately weak zero, 011 = weak zero, 100 = weak one, 101 = moderately weak one, 110 = moderately strong one, 111 = strong one). This representation was found to result in the simplest overall implementation (if the actual demodulator output assumes some other form, a simple format translator could easily be provided).

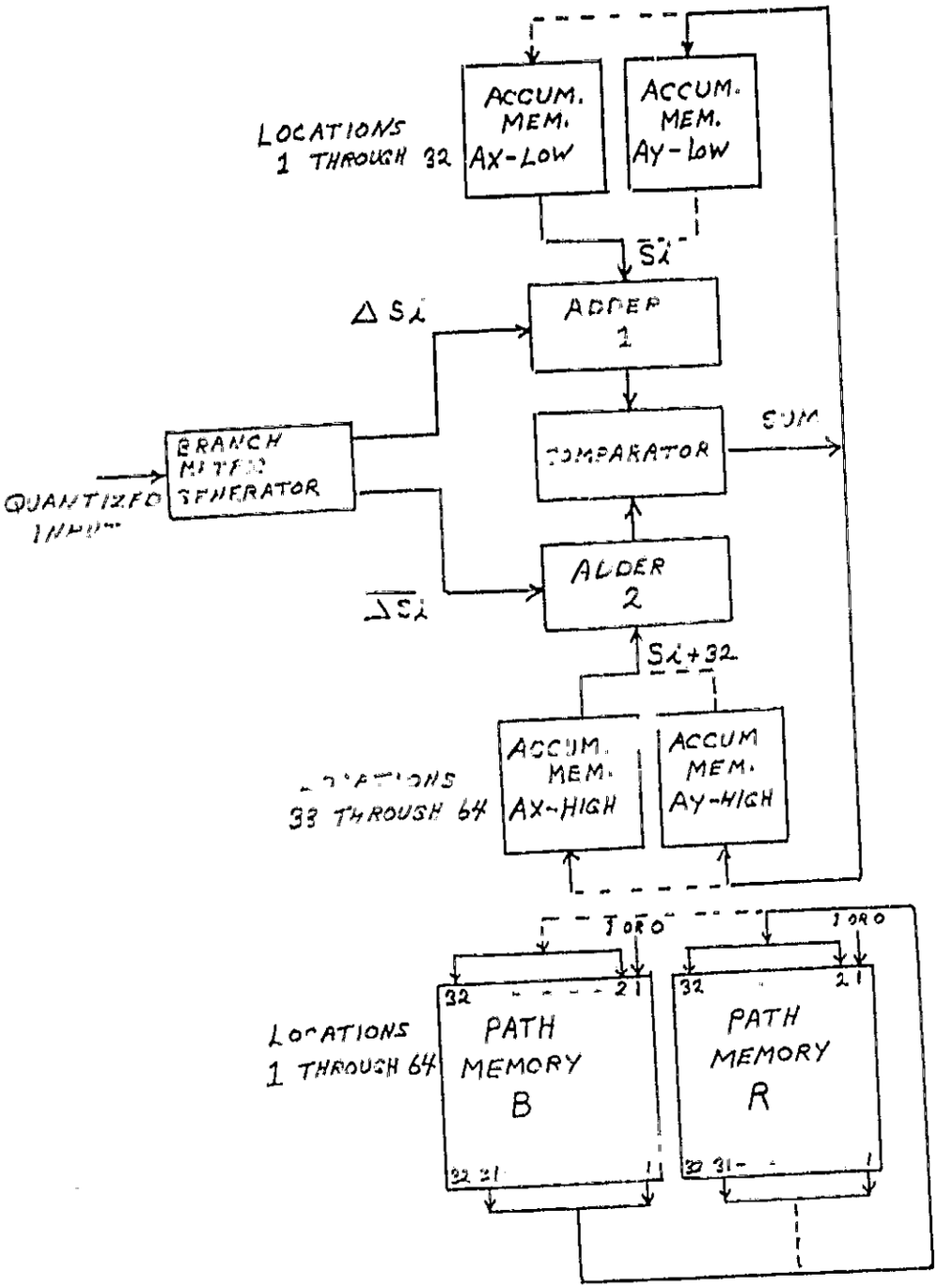
Once the branch-metric increments are generated, the add-compare-select (ACS) circuit updates the metrics associated with each surviving path, compares each of

*The algorithm described in this section is particularized to those codes which, like the one of interest here, have generators both beginning and ending with the all-ones V-tuple, with 1/V the code rate.



VITERBI ALGORITHM FLOW CHART

Figure 2.1



ORIGINAL PAGE IS
OF POOR QUALITY

SIMPLIFIED BLOCK DIAGRAM

Figure 2-2

these metrics with that of its competitor, and selects the one with the smallest metric. As indicated in Figure 2-1, the branch-metric increments ΔS_1, and $\Delta \overline{S}_1$,* for example, are added to the sums S_1 and S_{33} which are stored in accumulation memories Ax-low and Ax-high (cf. Figure 2-2). (Accumulation memories Ax-low and Ax-high and Ay-low and Ay-high are four separate memories with individual address selection. Dotted lines indicate that memory utilization alternates with each cycle of decoder operation, a cycle ending with the generation of an information bit (i = 32). Memories Ax-low and Ay-low contain locations one through 32, while memories Ax-high and Ay-high contain locations 32 through 64.) A comparison is performed on the outputs of the two adders and the lesser sum is stored in memory Ay-low, location 1.

At the same time, the contents of the path memory are modified as follows: If $S_1 + \Delta S_1$ is the lesser of the competitive metrics, path memory location B_1 is read and the output from bits 1 through 31 are loaded into bits 2 through 32 of duplicate path memory location R1, and a logic "0" is written into bit position 1. If $S_{33} + \Delta S_1$ is the lesser, path memory location B_{33} is read and the output from bits 1 through 31 are loaded into bits 2 through 32, and a logic "1" is written into bit 1 of the duplicate path memory location R1. (cf. Figure 2-2. As with the accumulation memories, the dotted lines here indicate that memory utilization alternates with each cycle of decoder operation.)

The next step in the ACS unit is to interchange the roles of ΔS_1 and $\Delta \overline{S_1}$. That is, S_1 is added to $\Delta \overline{S_1}$ and S_{33} is added to ΔS_1 with the two sums compared as before. If $S_1 + \Delta \overline{S_1}$ is less than or equal to $S_{33} + \Delta S_1$, then it is stored into Ay-low, location 2. Also, the contents of path memory location B1 (bits 1-31) are stored in location R2 (bits 2-32) and bit 1 is loaded with a logic "0". If $S_{33} + \Delta S_1$ is the lesser, then that sum is stored in Ay-low, location 2, and the contents of path memory location B33 are stored in location R2, bits 2 through 32, with a logic "1" loaded into bit 1.

---

*The terms ΔS_i and $\Delta \overline{S}_i$ here represent the metric increments corresponding to complemented branch patternsn; i.e., if ΔS is the metric increment when the received data are compared to the pattern xy = , the $\Delta \overline{S}_i$ is that obtained where these same data are compared with $\overline{x}\ \overline{y}\ \overline{z}$.

The identical sequence takes place for the remaining values of i, using the branch-metric increments ΔS_i and $\Delta \overline{S_i}$ with the locations in the metric and path memories altered accordingly. Finally, the output information bit is determined by comparing the number of "1s" and "0s" in the most significant bit (position 32) of each memory location just written into, with the majority state selected for the output bit.

Although the flow chart in Figure 2-1 indicates a serial operation with the index processing sequentially from 1 to 32, the decoder actually performs the 64 operations thus represented (two operations for each value of i) in eight steps of eight operations each. This is done to achieve the desired decoding rate (up to 100,000 information bits per second) while still using low-power CMOS logic. Furthermore, in order to simplify the metric calculator, these operations are so arranged that only one pair of metric increments, ΔS_i and $\Delta \overline{S_i}$, need be determined for any one of these steps. A related modification is also made in the ACS and path memory, units where the storage locations are shuffled (relative to those indicated in Figure 2-1) to simplify the address generators needed to identify all those locations corresponding to a given metric-increment pair.

The branch (or node) synchronizer implemented in the Viterbi Decoder brassboard was motivated by an examination of the null-space of the Odenwalder K=7, rate 1/3 convolutional code. It can be verified that any 3ℓ-symbol error-free code sequence $\underline{v}$ of this code is orthogonal to the $(2\ell - 6) \times 3\ell$ matrix

$$H = \begin{bmatrix}
011 & 010 & 110 & & & & & & & & & & & & \\
110 & 100 & 000 & 110 & 101 & & & & & & & & & & \\
 & 011 & 010 & 110 & & & & & & & & & & & \\
 & 110 & 100 & 000 & 110 & 101 & & & & & & & & & \\
 & & & & & \cdot & & & & & & & & & \\
 & & & & & & \cdot & & & & & & & & \\
 & & & & & & & \cdot & & & & & & & \\
 & & & & & & & 011 & 010 & 110 & & & & & \\
 & & & & & & & 110 & 100 & 000 & 110 & 101 & & & \\
 & & & & & & & & 011 & 010 & 110 & 000 & & & \\
 & & & & & & & & & 011 & 010 & 110 & & & \\
 & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & &
\end{bmatrix}$$

That is, $H\underline{v} = \underline{0}$ for all sequences generated by the convolutional encoder. This statement is true, however, only if the sequence $\underline{v}$ begins at a node in the code tree. If $\underline{v}$ begins with either of the two other symbols in a code tree branch, in contrast, and if the encoder input sequence is random, $H\underline{v}$ is a random sequence of ones and zeros.

Note that rows two and three are orthogonal on two bits. This property is exploited as follows: The sign bit from each decoder input is read into a 15-bit auxiliary shift register and the modulo-two sum formed of the contents of the register's 1st, 3rd, 12th, 14th and 15th stages, a second modulo-two sum formed of the contents of its 8th, 10th and 11th stages, and a third modulo-two sum formed of its 5th and 6th stages. If the shift-register's contents are properly framed and if the polarity of its inputs is correct, all three parity checks will agree, at least in the absence of errors. (I.e., the first and third of these parity checks correspond to the second row of H and the second and third to the third row of H.) If the framing is correct but the polarity reversed, the first and second parity check will agree, again, at least in the absence of errors, and will equal the complement of the third parity check (since the latter is the only one based on an even number of bits). If the framing is incorrect, however, it is easily demonstrated that the three parity bits will be randomly related. (Note that all vectors in H are cyclic shifts of the two vectors actually used. Consequently, all code information available to the Viterbi decoder is used in the synchronizer, although, of course, much less thoroughly.)

The results of these parity-check comparisons can, therefore, be used as an indicator of correct or incorrect node synchronization. If, in particular, all three parity checks agree, an up-down counter is incremented; if the first two agree but disagree with the third, the same up-down counter is decremented; in any other situation, the counter is neither incremented nor decremented. The process is repeated when the next bit is read in to the shift-register but with a second up-down counter, and again with the third input bit and a third up-down counter. The whole sequence begins again with the fourth input so that, in general, the i^{th} input causes the j^{th} up-down counter to be incremented or decremented with $j = i \pmod 3$. It is readily verified that the up-down counter associated with the properly framed code sequence is more likely to be augmented (decremented) and less likely to be decremented (augmented) when the polarity is correct (reversed) than are either of the other two counters, even in the presence of errors. Correct node synchronization is thus estimated by observing

which of the counters first reaches a predetermined positive or negative threshold T. Further, a negative threshold crossing indicates an inverted decoder input so that all node synchronization and sign ambiguities are resolved simultaneously.

2.2 General Implementation

A complete block diagram of the decoder is shown in Figure 2-3. The 3-bit quantized demodulator output is received by the Metric Calculator logic where the branch metric calculations are performed. The resulting 5-bit metric increment numbers (Δ S) are used two at a time in the add, compare, select (ACS) arithmetic section. This arithmetic section is divided into eight identical units, each of which performs eight ACS operations in one bit-time period.

The results of the 64 decisions made in the ACS unit are sent to the path memory in the output selector section. The path memory is divided functionally into eight indentical groups of 8 words by 32 bits in a one-to-one correspondence with the ACS metric memory. During one information-bit period, data are read from one memory and stored in a second duplicate memory. The memory roles are reversed during the following bit period; the second memory is read and a write performed into the first. The output bit is determined by comparing the number of "1s" with the number of "0s" in the oldest bit position for each path and selecting the one occurring more frequently.

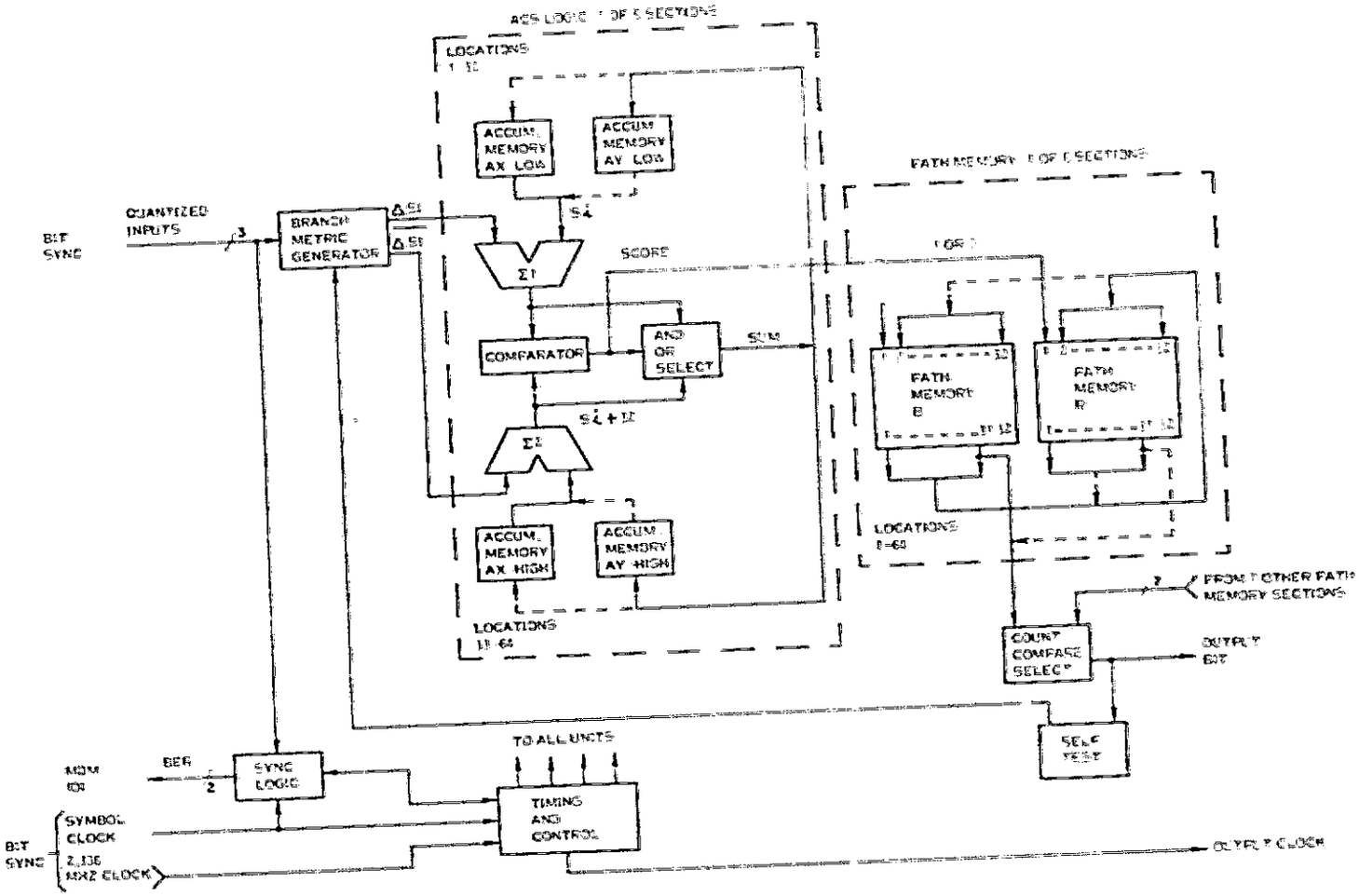
Synchronization logic receives the most significant bit of each input symbol and resolves from these the node and phase ambiguities.

In the self-test mode a fixed data pattern is presented to the metric calculator section and the resulting decoder output compared with an appropriately delayed version of the same pattern.

Finally, a timing and control section provides the sequence, sub-sequence, control functions and output clocks for the decoder.

Each of these decoder subunits is described in detail in the following sections.

ORIGINAL PAGE IS
OF POOR QUALITY



VITERBI DECODER BLOCK DIAGRAM

Figure 2-3

2.2.1 Metric Calculator, Node Synchronizer and Control Logic

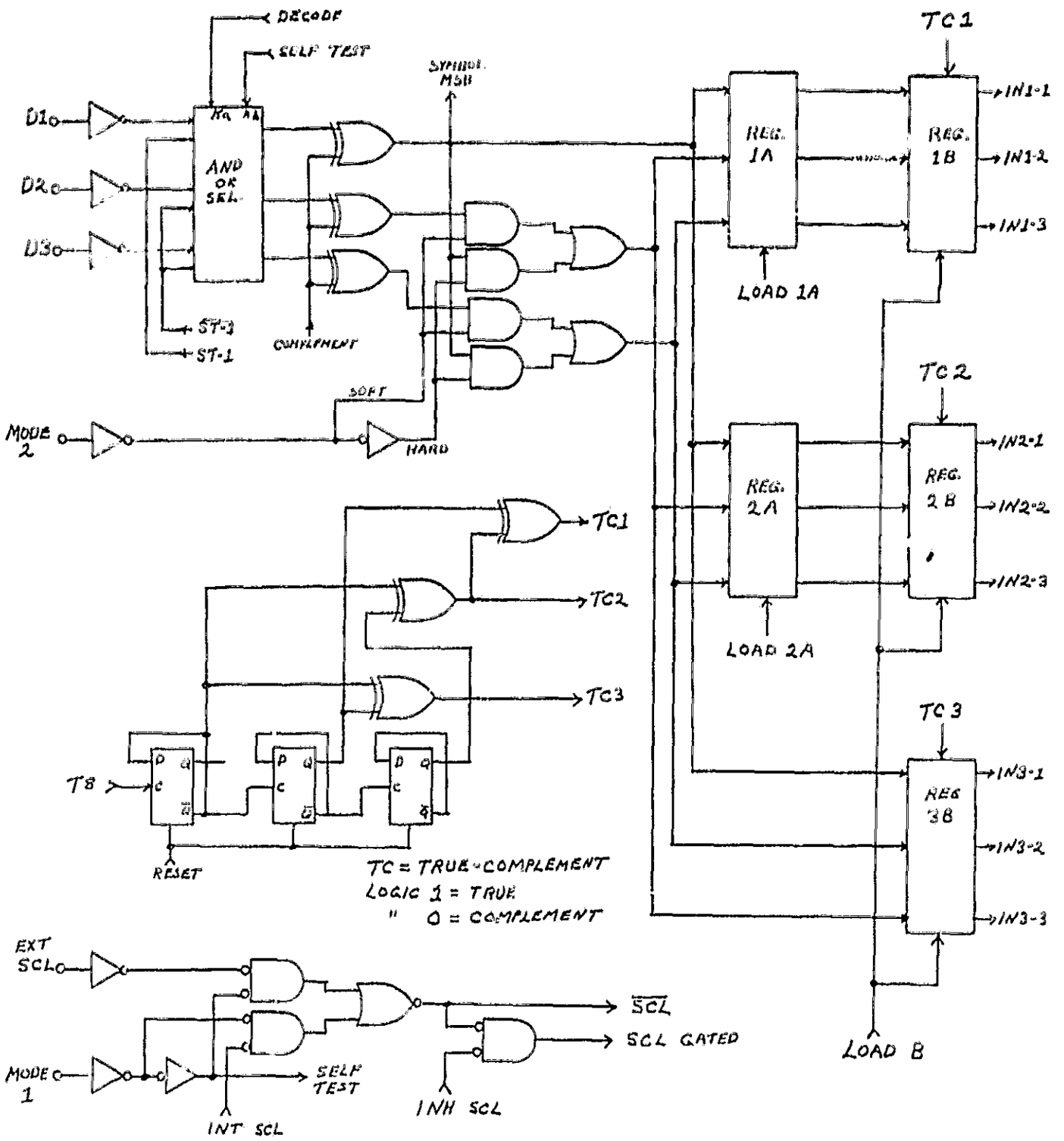
A block diagram of the metric calculation logic is shown in Figures 2-4 and 2-5. Detailed logic for this section is shown on drawing VDBD#1, Sheets 1 and 4.

Input symbols are received at the decoder with three DM88L12, low power TTL, level translator gates which receive the 5 volt interface signals and output them at the 10 volt CMOS logic levels. The translator outputs are fed into an and-or-select (AOS) unit which provides selection of either the normal input symbols or the internally generated self-test symbols. Each output of the AOS unit is exclusive-ored with the complement signal which is generated by the node synchronization logic. In the soft-input mode all three inputs are passed on to the storage registers; in the hard-input mode, the most significant symbol bit only is used for all three inputs. The first two symbols are loaded into buffer registers 1A and 2A, while the third symbol is loaded directly into holding register 3B. At the time symbol #3 is loaded into 3B, the contents of registers 1A and 2A are transferred into holding registers 1B and 2B, respectively. This data transfer initiates the decode operation which is completed before the third symbol of the following group is received.

Data represented by the three 3-bit numbers in the registers are added in either the true or complemented form depending on the specific branch metric increment being determined. The output of the two adder circuits then represent the metric increments ΔS_i and $\Delta \overline{S}_i$.

Δ S Control Logic

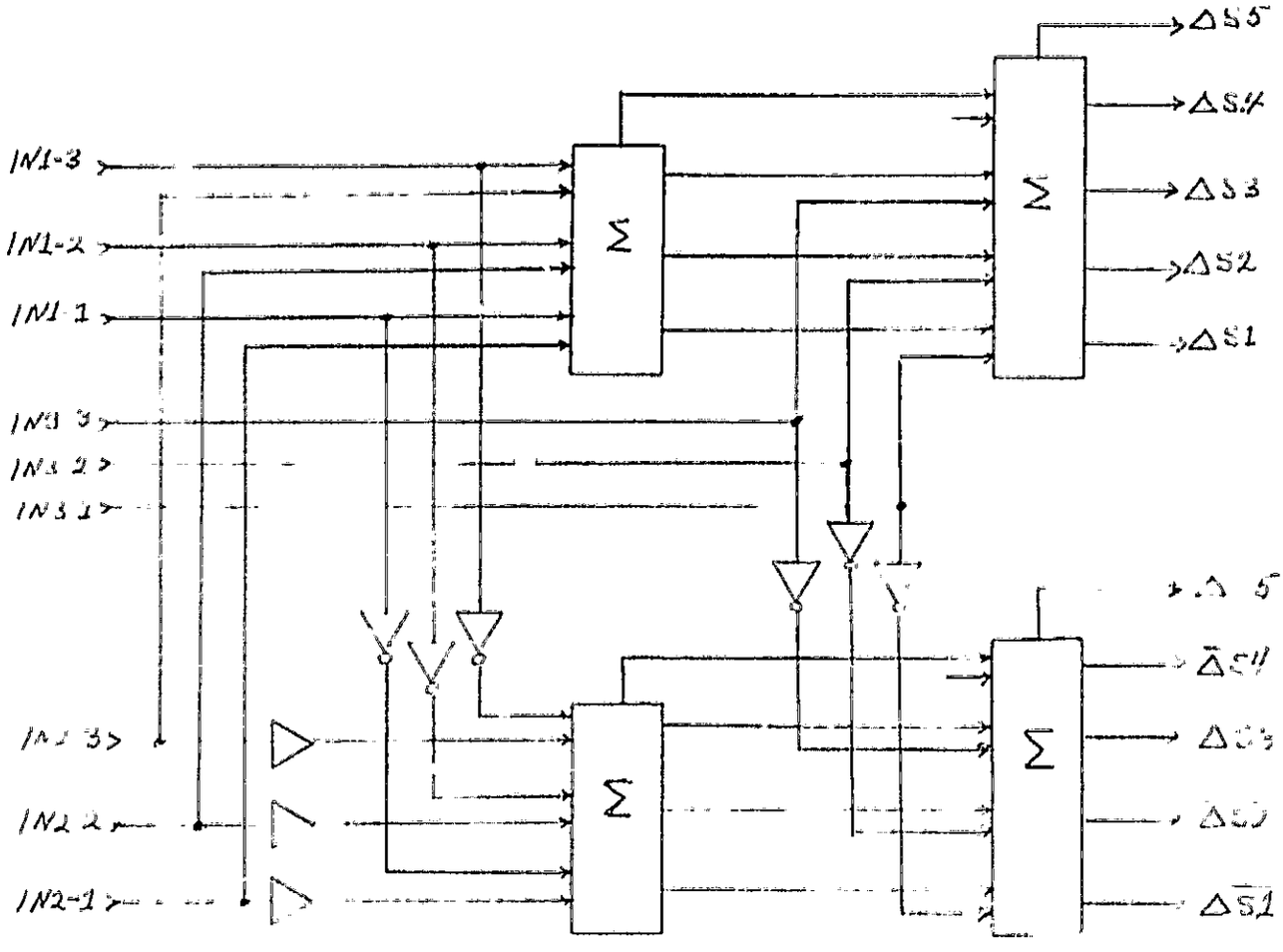
The logic shown on the left-center section of Figure 2-4 generates the true/complement control signals TC1, TC2 and TC3 for the registers 1B, 2B and 3B, respectively. The binary counter is incremented at eight times the output bit rate to generate the following control pattern:



INPUT LOGIC

Figure 2-4

ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY

METRIC CALCULATION LOGIC

Figure 2-5

1 - 000
2 - 111
3 - 101
4 - 010
5 - 110
6 - 001
7 - 011
8 - 100

A logic "0" provides a "true" output, and a logic "1" provides a "complement" output. With respect the the flow chart of Figure 2-1, the control function 000 yields the metric increments Δs_1 and $\Delta \overline{s}_1$. The next control 111 yields the same quantities, $\Delta \overline{s}_1$ and Δs_1, but in reversed order, etc.

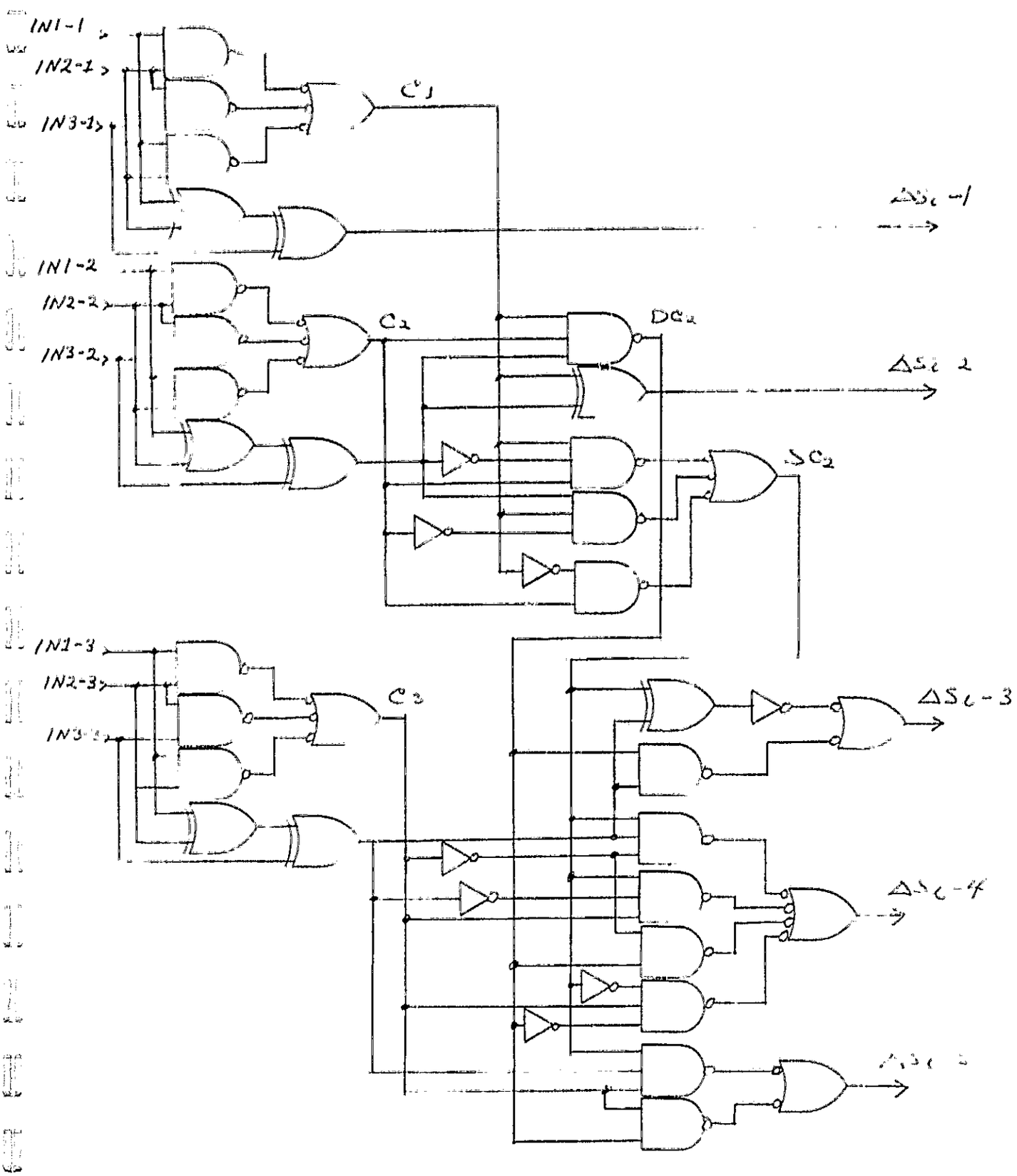
Logic in the lower-left of Figure 2-4 is for symbol clock select. In the decode mode, the external symbol clock is level-translated and selected for use as the symbol load clock. An internally generated symbol clock is selected in the self-test mode.

Three-Number Adder

Logic for the three-number adder is shown in Figure 2-6. Inputs from the three holding registers are added to generate the five-bit branch-metric increment, Δs_i. An identical circuit receives the inverted branch-metric increment, $\Delta \overline{s_i}$. The small scale integrations(SSI) logic used for the adders is of silicon-on-sapphire (SOS) technology made by Inselek Co.

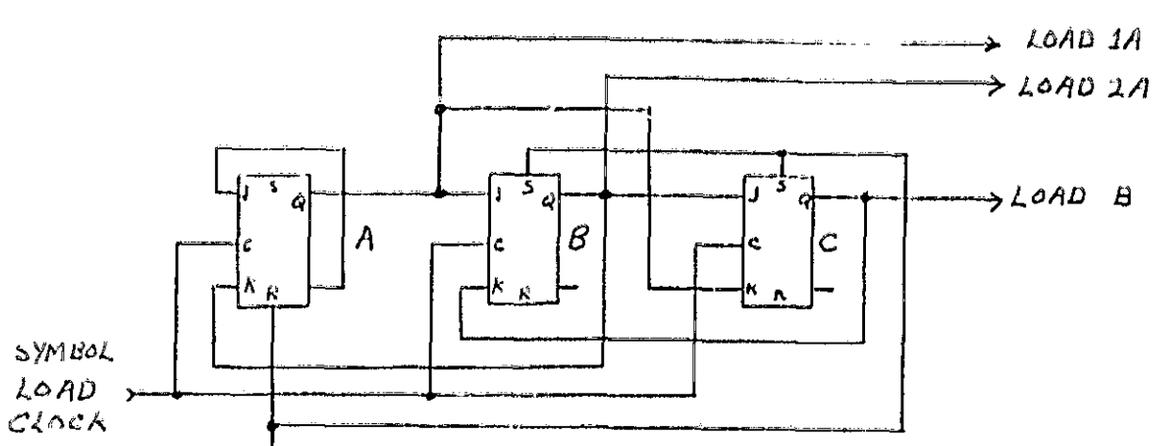
Symbol Load Control Signal Generation

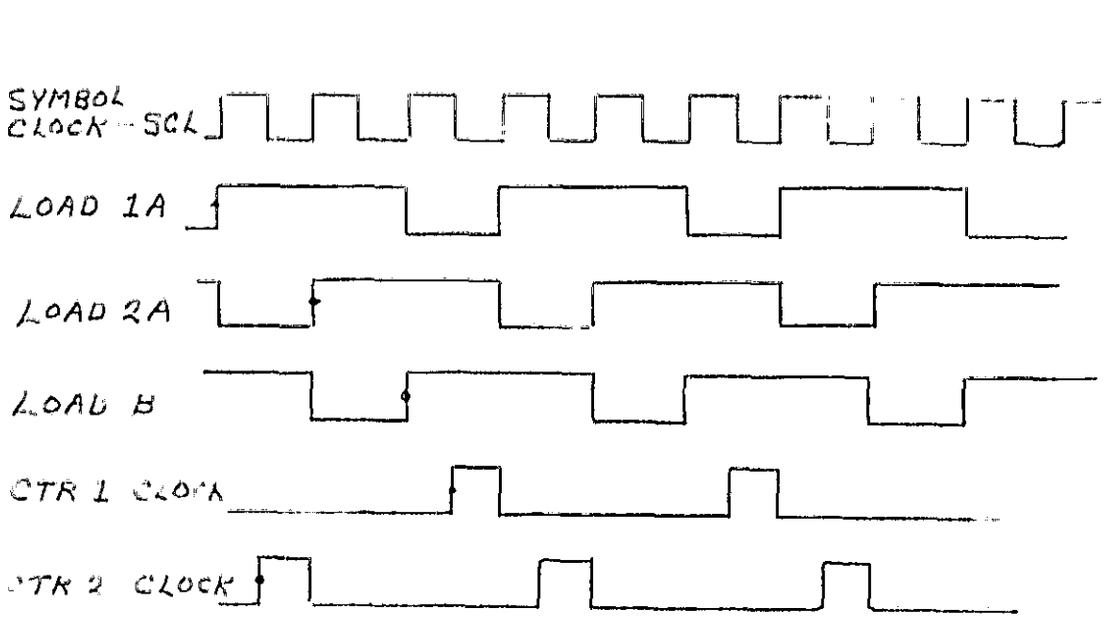
Signals for loading the two-buffer registers 1A and 2A, and the three holding registers 1B, 2B and 3B are generated by the logic of Figure 2-7. From these signals are derived the signals for clocking the three up-down counters in the node synchronization logic.



THREE NUMBER ADDER

Figure 2-6





SYMBOL LOAD SIGNAL GENERATION

Figure 2-7

Decoder Output Clocks

Output clocks at 0° and 180° are provided at the information bit rate. The Load 2A signal is buffered into the 9614 driver gates connected to provide the two clock phases (VDBD #1, Sheet 1). The clock duty cycle is one-third.

Node Synchronization Logic

Node synchronization logic is comprised of the following sections:

Node-sync shift register, node-sync up/down counters, enable sync-change logic and node sync control logic. Detailed logic is shown on Drawing VDBD #1, Sheets 2 and 3.

Node-Sync Shift Register (Figure 2-8)

The 15-bit node-sync shift register receives each symbol MSB at the symbol clock rate. Selected register outputs are exclusive-ored to produce three signals A, B and C. From these signals are then generated the up/down count, and inhibit-count control signals.

Node-Sync Up/Down Counter (Figure 2-9)

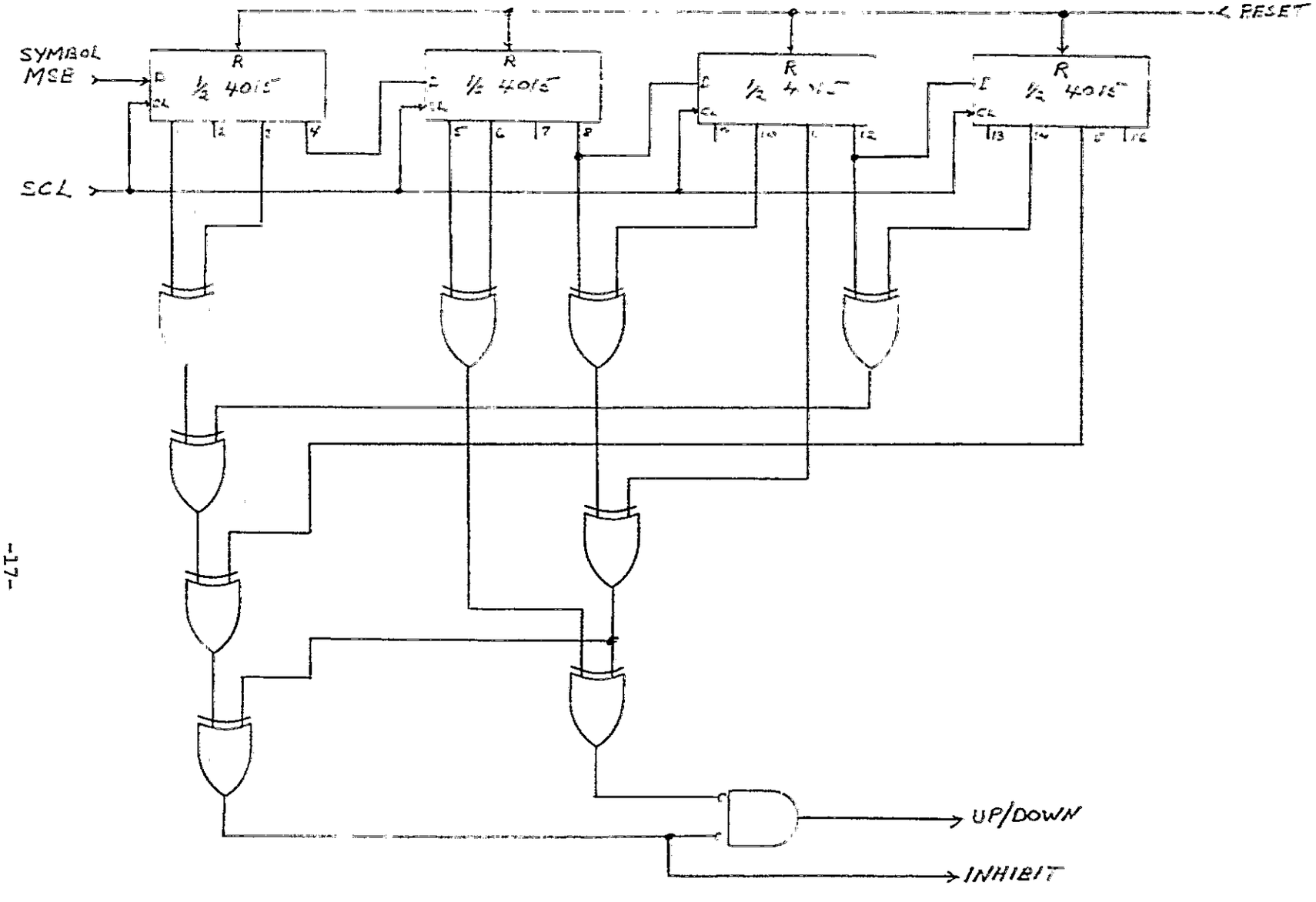
Three 8-bit binary up/down counters are required for the node synchronization implementation. Each 8-bit counter consists of two CD4029 4-bit up/down counter units. The counters generate overflow signals at the count of 64 when counting up, and generate underflow signals at count of 0 when counting down. When any one counter overflows or underflows, all three are then reset to a count of 32.

Node-Sync Control Logic

The synchronization action to be taken is determined according to the overflow or underflow indication from a counter, as follows:

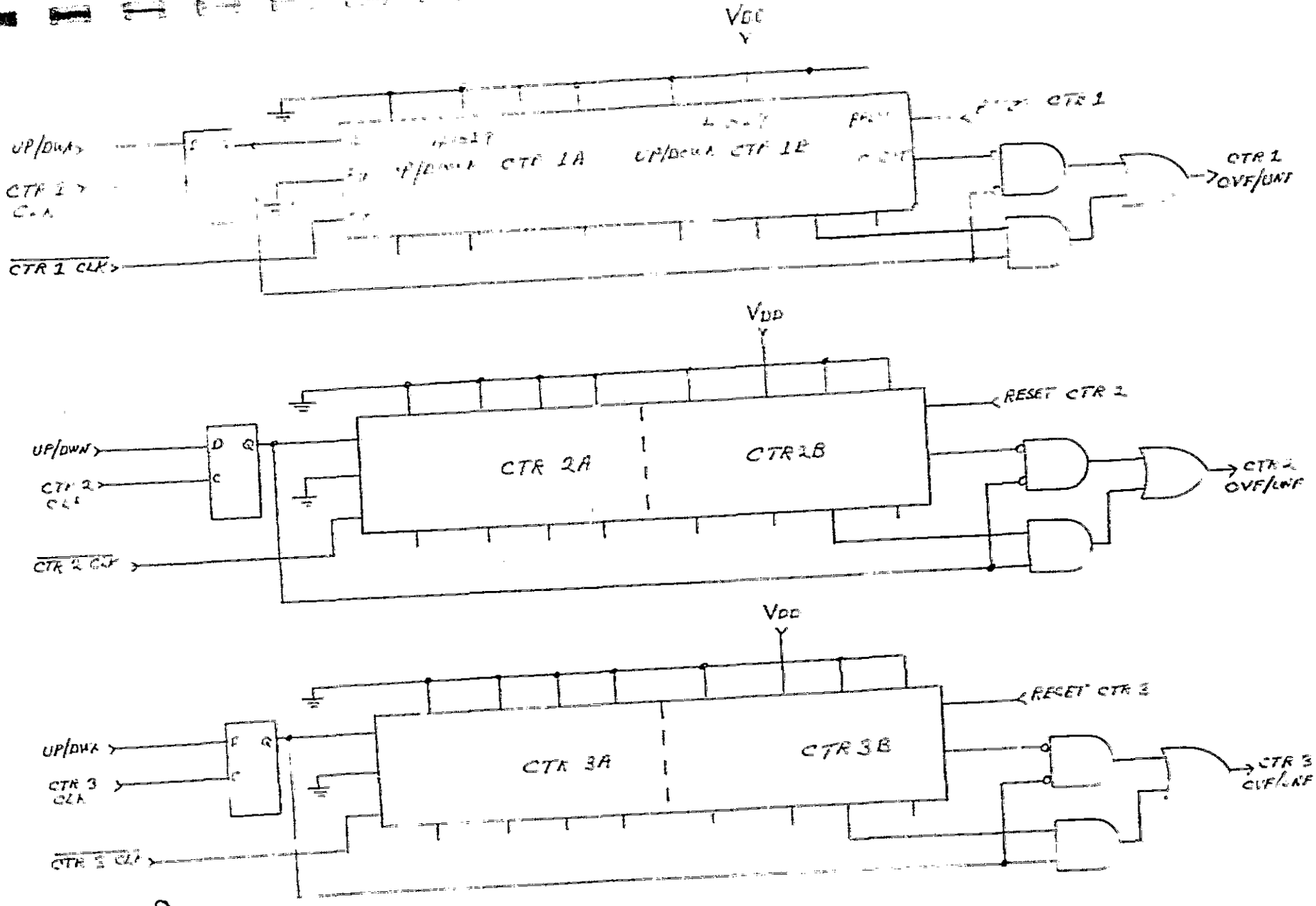
Counter #	Overflow	Underflow
1	Sync OK	Complement Symbol
2	Slip 1 Symbol	Slip 1 Symbol and Complement
3	Slip 2 Symbols	Slip 2 Symbols and Complement

Logic for implementing these functions is shown in Figure 2-10. A symbol is slipped by inhibiting the symbol load clock for one or two periods, as required. An overflow or underflow from any counter following start-up will set the in-sync flip-flop shown at the bottom of Figure 2-10 and drive an In-Sync indicator in the



NODE-SYNC SHIFT REGISTER

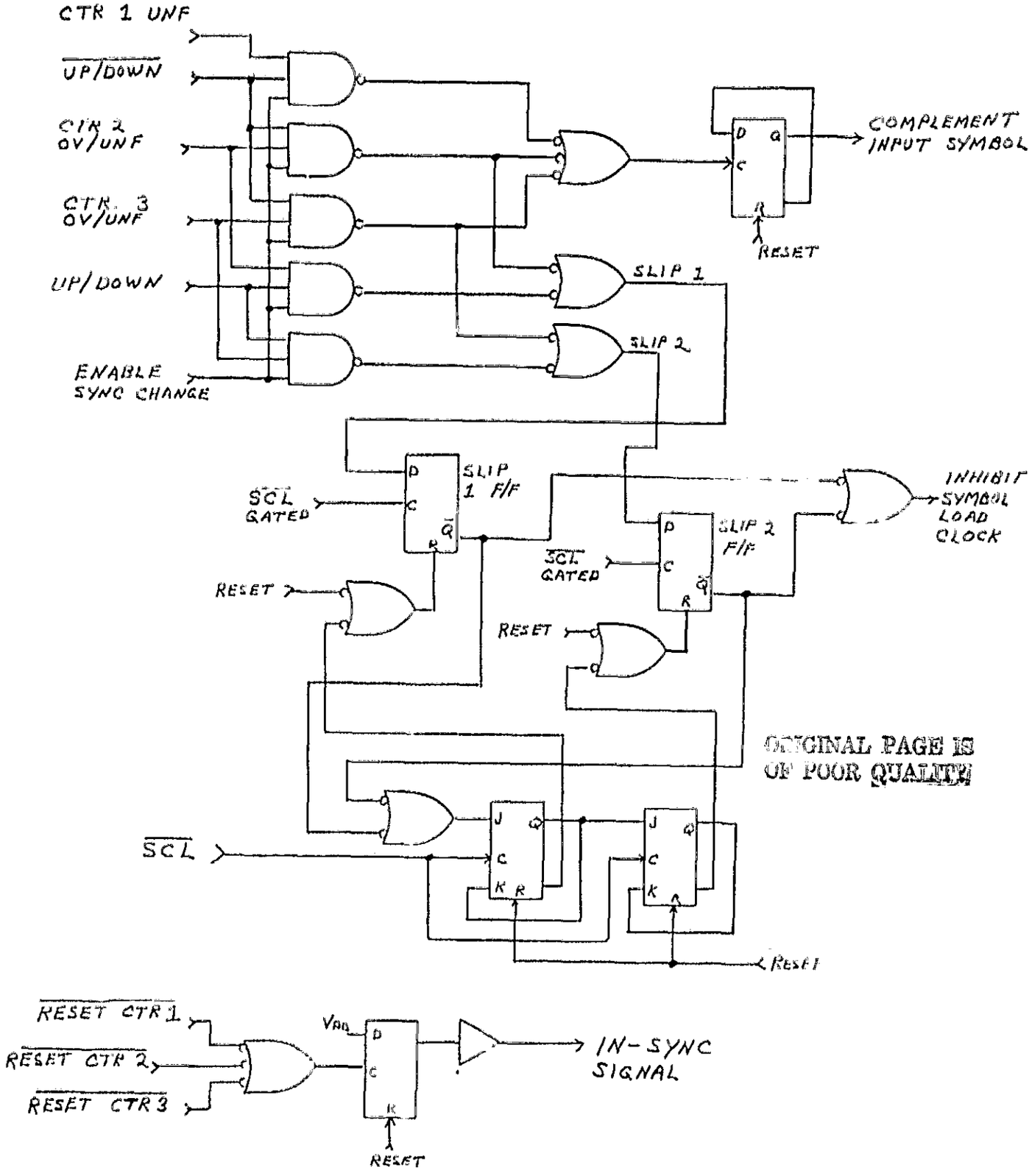
Figure 2-8



NODE SYNC UP-DOWN COUNTERS

Figure 2-9

ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY

NODE SYNC CONTROL LOGIC

Figure 2-10

test set.

The clocks for the three up/down counters are inhibited for sixteen symbol clock times following a change in sync. This allows the sync shift register to refill with symbols in the correct sequence before any counters are incremented or decremented. Logic for this function is shown in Drawing VDBD #1, Sheet 2.

Enable Sync-Change Logic

Figure 2-11 is a block diagram of the logic required to generate the enable sync-change signal. Detailed logic is located on Drawing VDBD #1, Sheet 3.

At decoder start-up, the Enable Sync F/F forces ENSYNC CHANGE signal to the true state until either a complement signal or an inhibit symbol load clock signal is received.

A four stage up/down counter, EE9 controls resynchronization following start-up. The counter is initialized to the count of one and counted up or down according to the node sync up/down counter outputs. If counter #1 overflows, indicating correct sync, EE9 is incremented twice. A counter #1 underflow, or a counter #2 or #3 underflow or overflow decrements EE9. When the count of sixteen is reached any additional up-counts are inhibited. If the counter is decremented to zero, the signal RESYNC is generated which in turn activates the ENSYNC CHANGE signal allowing decoder resynchronization.

A resynchronization test point which can drive a test set indicator is generated by flip/flops DB43 and DF44. The first ENSYNC CHANGE signal transition at start-up is trapped in DB43 which enables flip/flop DF44. The next ENSYNC CHANGE signal transition will be trapped in DB43 and the RESYNC test point energized.

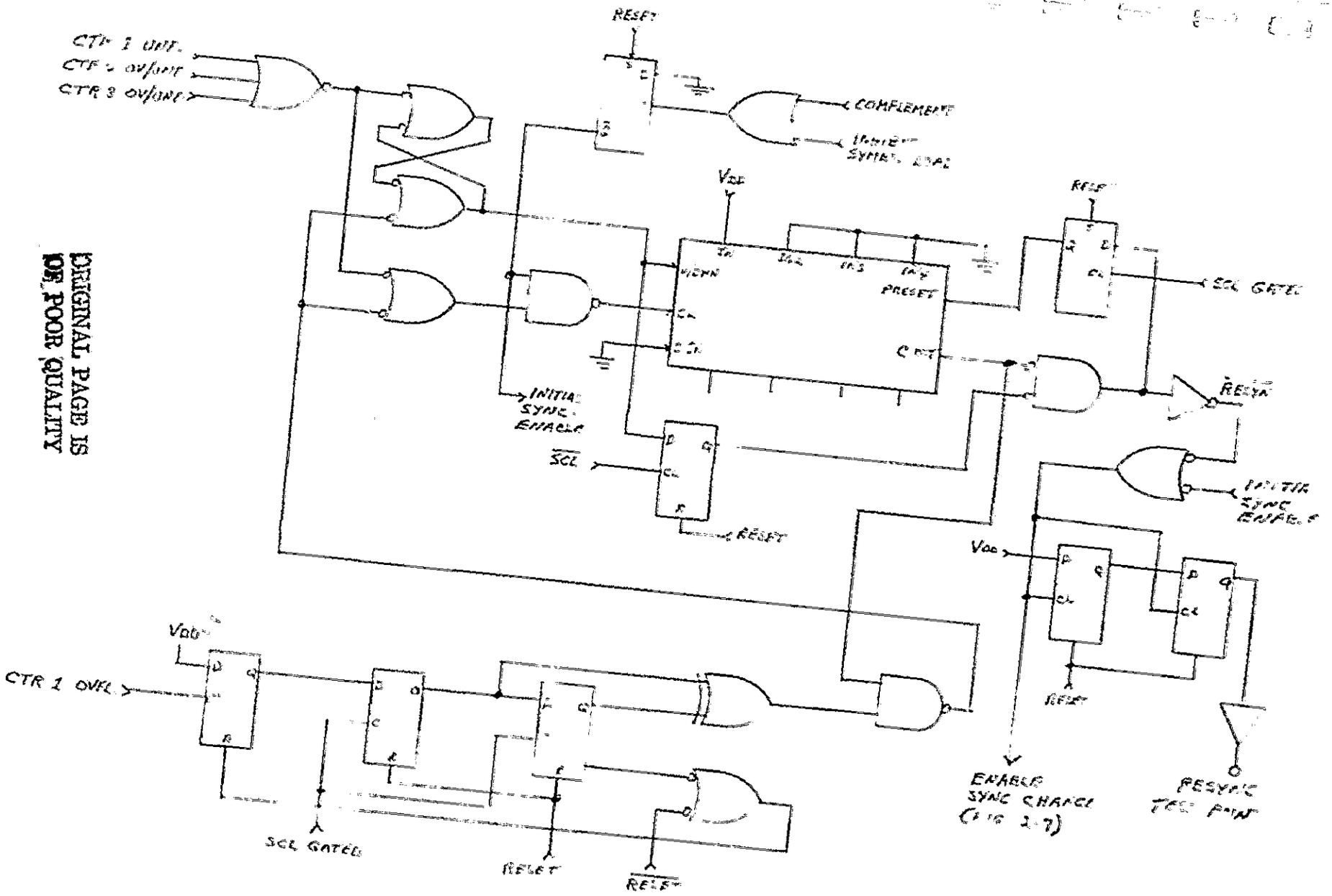
Decoder Input Timing

Timing for input symbol load signals and node synchronization counter increment and control signals is shown in Figure 2-12. All time intervals in this section are derived from the symbol clock which is variable according to the input symbol rate.

Self-Test Symbol Generator

The decoder self-test concept requires the generation of a pseudo-random bit pattern which is encoded and presented to the decoder in 3-bit symbols. The decoded output is then compared with the output of the pattern generator shift register. A block diagram for the pattern generator and encoder is shown on

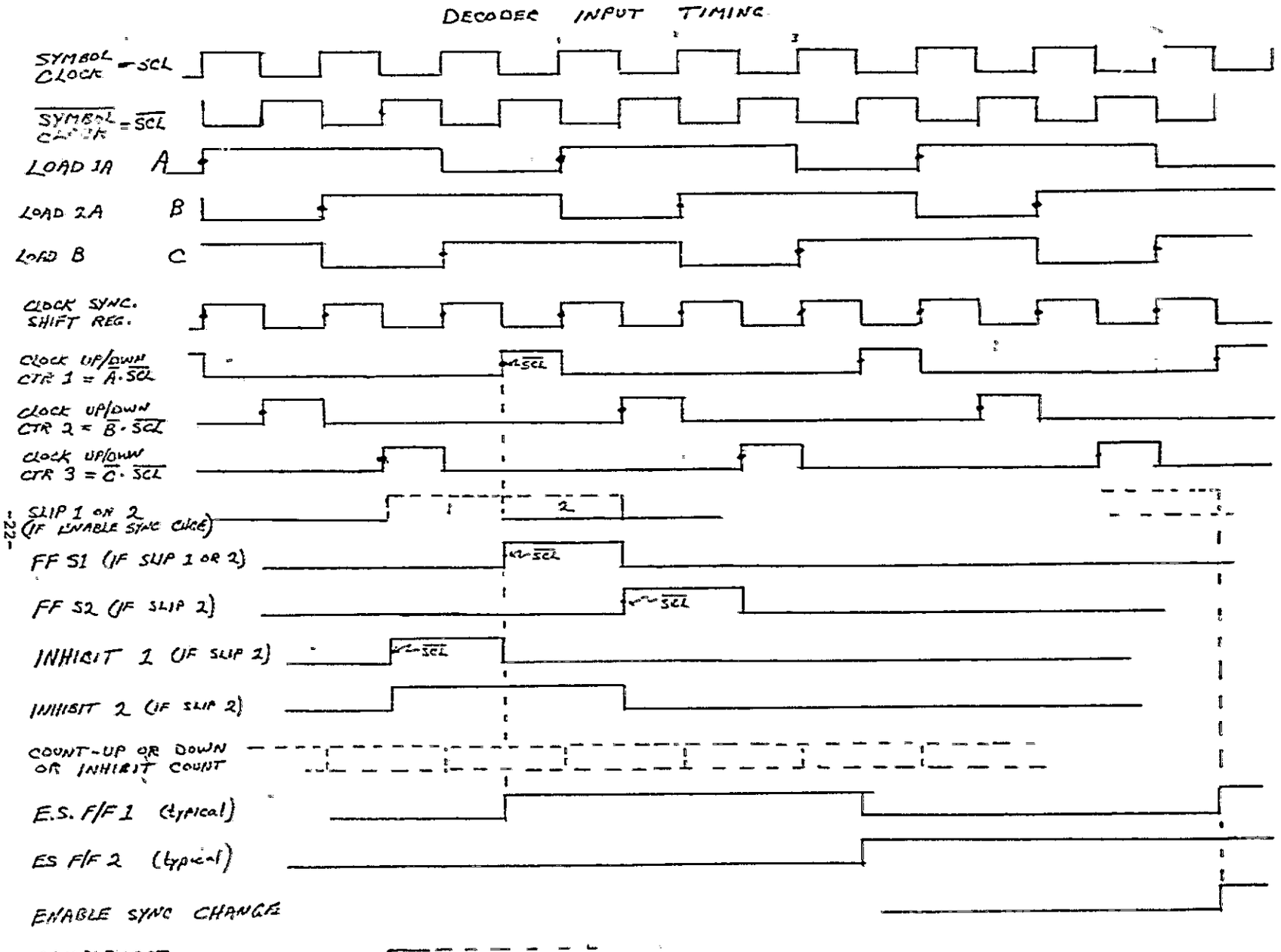
ORIGINAL PAGE IS
OF POOR QUALITY



ENABLE SYNC-CHANGE LOGIC

Figure 2-11

ORIGINAL PAGE IS
OF POOR QUALITY



DECODER INPUT TIMING

Figure 2-12

Figure 2-13. Detailed logic is located on Drawing VDBD #1, Sheet 3.

The pattern generator is a 7-stage convolutional encoder shift register with the modulo-two sum of the contents of the third and fifth stages fed back to generate a pseudo-random sequence of length 31. The diagram shows two additional stages added to the shift-register serial output to provide the 33-bit delay corresponding to that through the decoder with a path memory length of 32 bits.

Self-Test Control Logic (Figure 2-14)

The transition from decode mode to self-test mode initiates a Reset 1 pulse which resets the entire decoder. An internal symbol clock is then generated at approximately 250 KHz to clock the self-test symbol generator and the symbol load control logic. The 250 KHz basic clock frequency is provided by an R/C oscillator circuit.

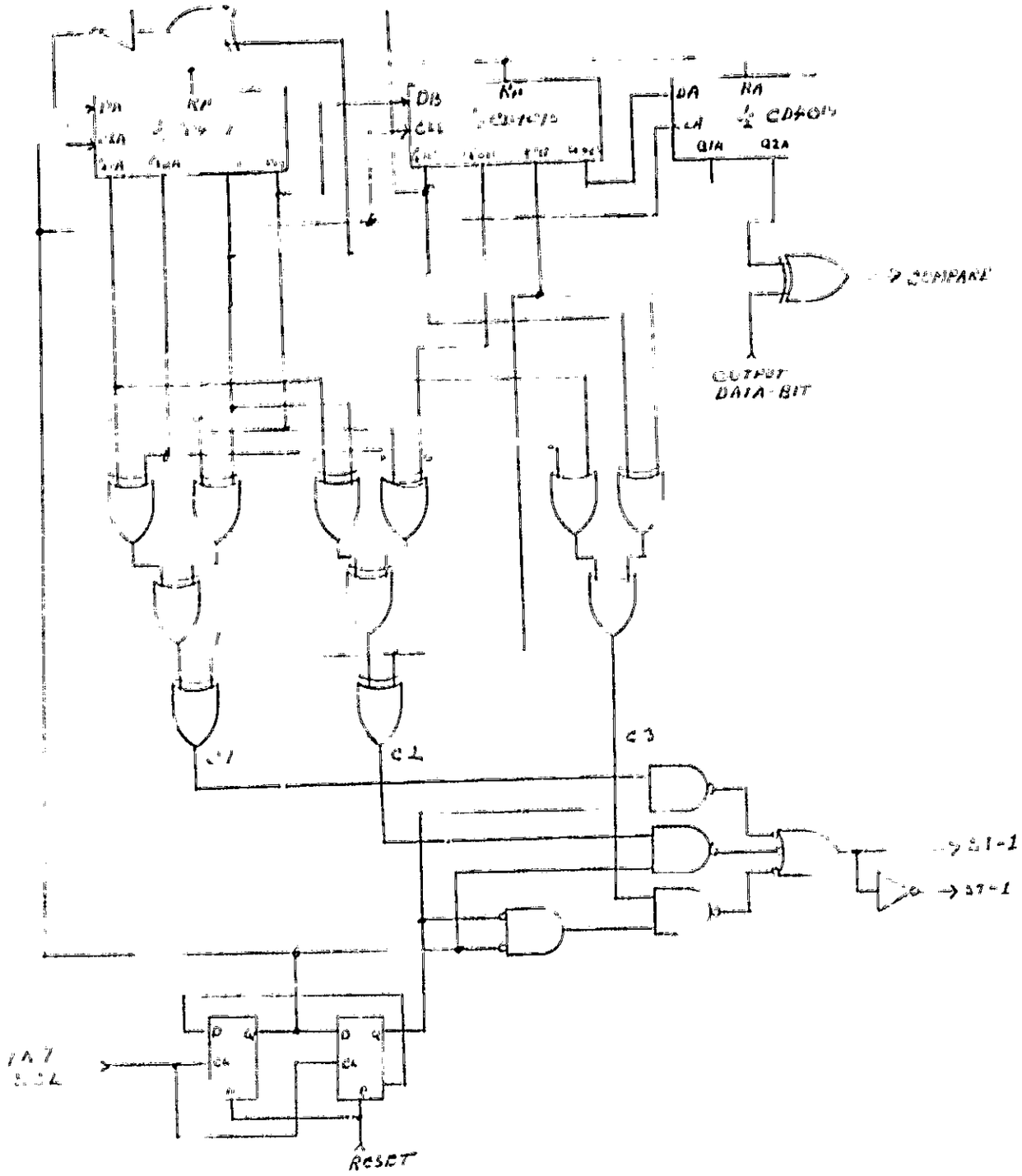
A bit delay interval at least equal to the path memory length plus sync acquisition delay is required before a valid output comparison can be made. Two CD4024 7-bit counters were chosen to provide a 128 bit delay.

Sequence Logic (Figure 2-15)

Sequencer timing for the decode operation is derived from a 6.5 MHz local crystal oscillator which is divided by 8 to produce the 154 nsec. time pulses, T1 through T8. These pulses are in turn divided by 8 to generate sequence pulses 1 through 8, each of 1.23 μsec duration. Logic selected for the sequencer is Inselek SOS CMOS which operates at the required frequencies without excessive propagation delays. Detailed logic is shown on Drawing VDBD #1, Sheet 3.

The sequencer is started upon receipt of the third symbol which enables FF ST1 to be set on the next positive transition of the local oscillator. The output of ST1 sets ST2, enabling the time pulse generator and allowing the counter to be advanced from T1 to T2 on the next rising clock edge. Time pulse generation continues through T8, then repeats. On the trailing edge of the T8 pulse, a carryout signal is generated which clocks the sequence generator from sequence 1 to sequence 2.

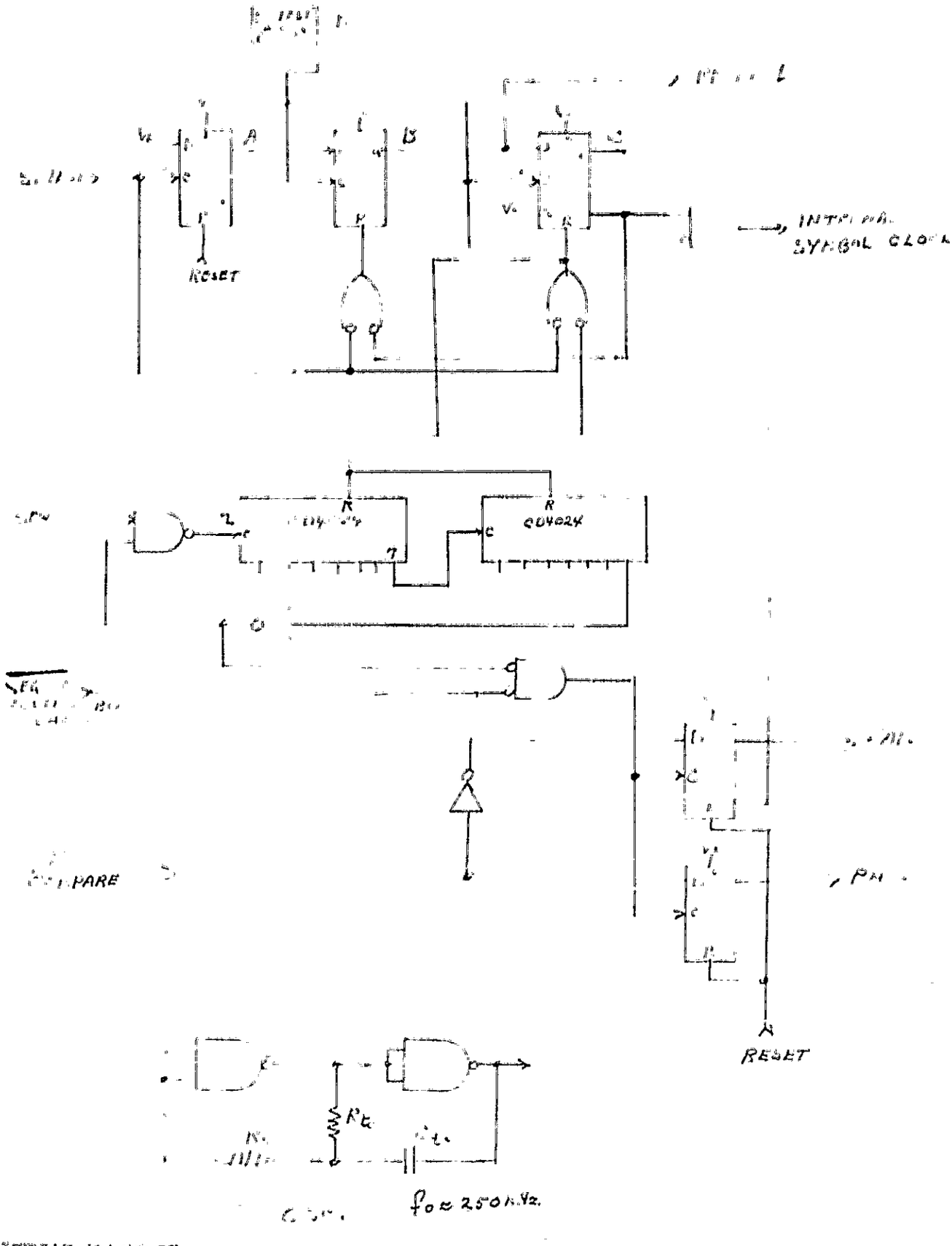
The end of sequence 8 indicates completion of the decode operation for one output bit. Flip-flop ST3 is set, ST2 is reset, and the sequencer operation is halted until the third symbol is received for the next information bit. The time required to synchronize the symbol-load clock the OSC signal before start of a new cycle is greater than was anticipated. The maximum bit rate for the decoder



SELF-TEST SYMBOL GENERATOR

Figure 2-13

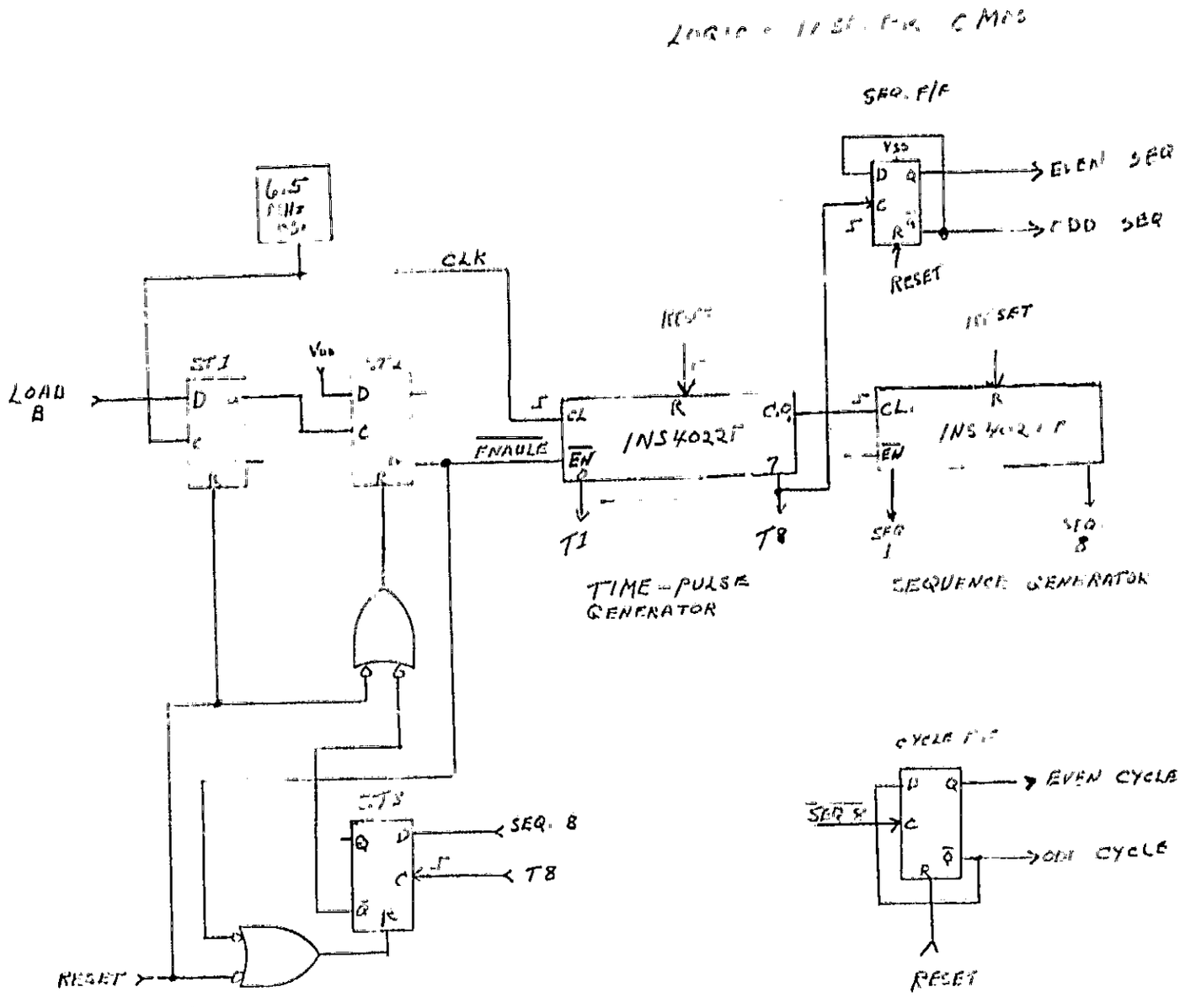
ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY

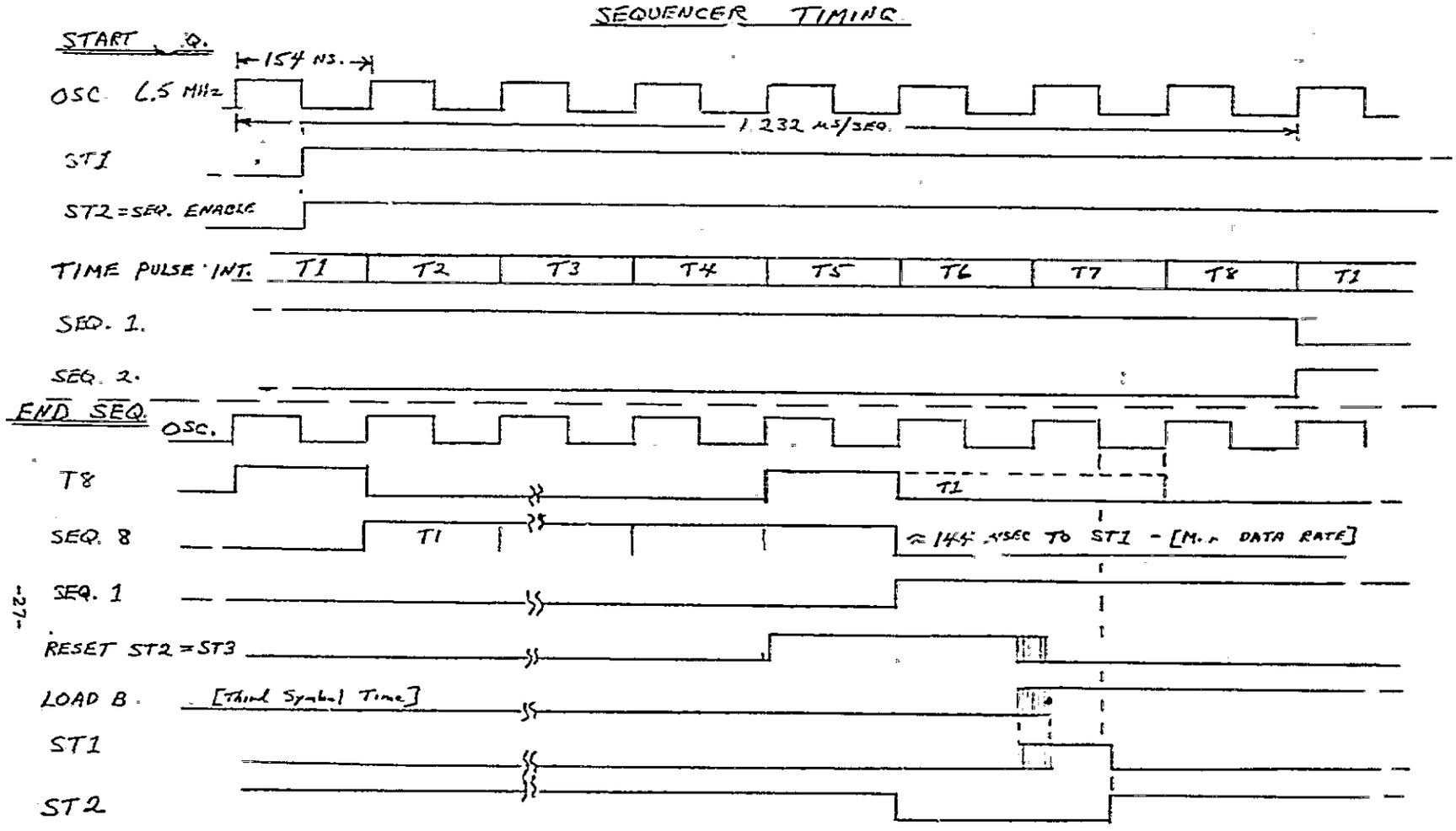
SELF-TEST CONTROL LOGIC

Figure 2-14



SEQUENCER LOGIC

Figure 2-15



NOTE: SEQUENCER STOPS IN SEQ. 1, T1, CONDITION

@ 6.5 MHz, 1 TIME PULSE = 154 NSEC,
1 SEQ. = 8 X 154 = 1.232 μSEC
1 CYCLE = 8 X 1.232 = 9.856 μSEC. ∴ ≈ 144 NSEC. FROM END OF SEQ. 8 TO ST1

SEQUENCER TIMING

Figure 2-16

is therefore limited to approximately 99 KBPS with the existing 6.5 MHz oscillator. The use of a slightly higher frequency oscillation would allow the sequence to be operated at the 100 KBPS rate. Sequencer timing is shown on Figure 2-16.

A sequence flip-flop provides the odd-even sequence indication for control signal generation. The flip-flop is reset to odd sequence and clocked to even sequence when the sequence generator advances to sequence 2.

An output-bit cycle indication is required for ACS and path memory read-write control signals. The cycle flip-flop shown on Figure 2-15 provides the odd-even cycle indication which changes every #8 sequence time.

ACS Control Signals

Detailed logic for controlling the ACS memory address increment, and for generating the read, write and normalization signals is shown on Drawing VDBD #1, Sheet 3.

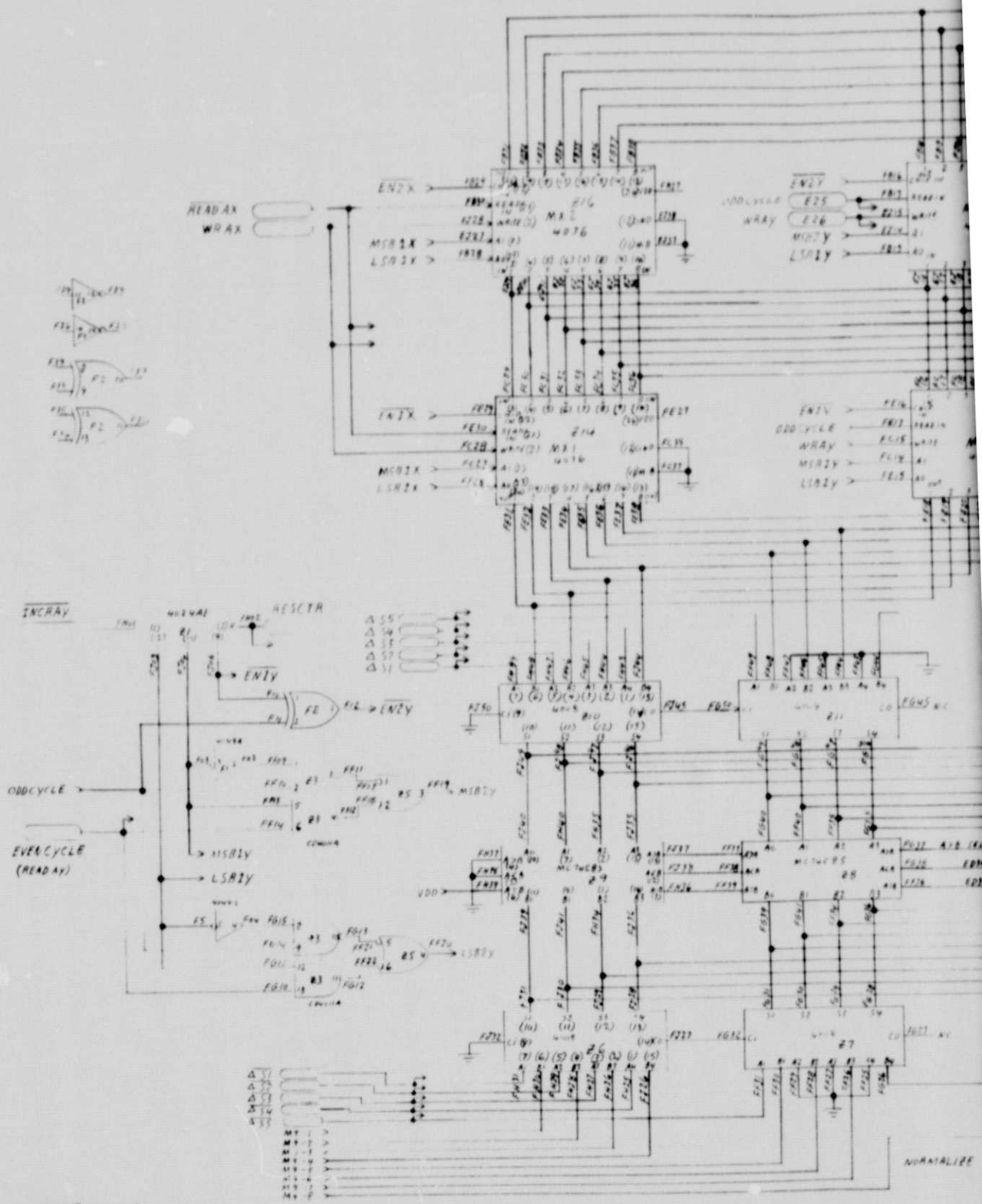
Path Memory Control Signals

Logic for the path memory address and read-write control signal generation is located on Drawing VDBD #1, Sheet 4.

2.2.2 ACS Logic

The logic for ACS Group #1 is shown in Figure 2-17, and is typical for each of the eight ACS groups. Detailed logic for each ACS group is shown on Drawings VDBD #1, Sheets 1 through 8.

The 5-bit number ΔS_1 is presented to the upper 8-bit adder, along with the 8-bit accumulation sum, S_1, read from MS1 memory. At the same time the 5-bit number $\Delta \overline{S}_1$ is added to the 8-bit sum, S_{33} (M9-1 through M9-8), read from the MX9 memory. The outputs of both adders are compared and the smaller sum is gated through the and/or select gates to MY1 memory location Ay1. The comparator result is also sent to path memory control for the first of eight path memory sections. This and analogous operations in each of the other seven sections complete the ACS action for the first of eight sequence intervals. During the subsequent sequence intervals, the above operations are repeated with, for example, $\Delta \overline{S}_1$ added to the same S_1 sum in the upper adder of the first section during the second interval, while ΔS_1 is added to S_{33} in the lower adder.

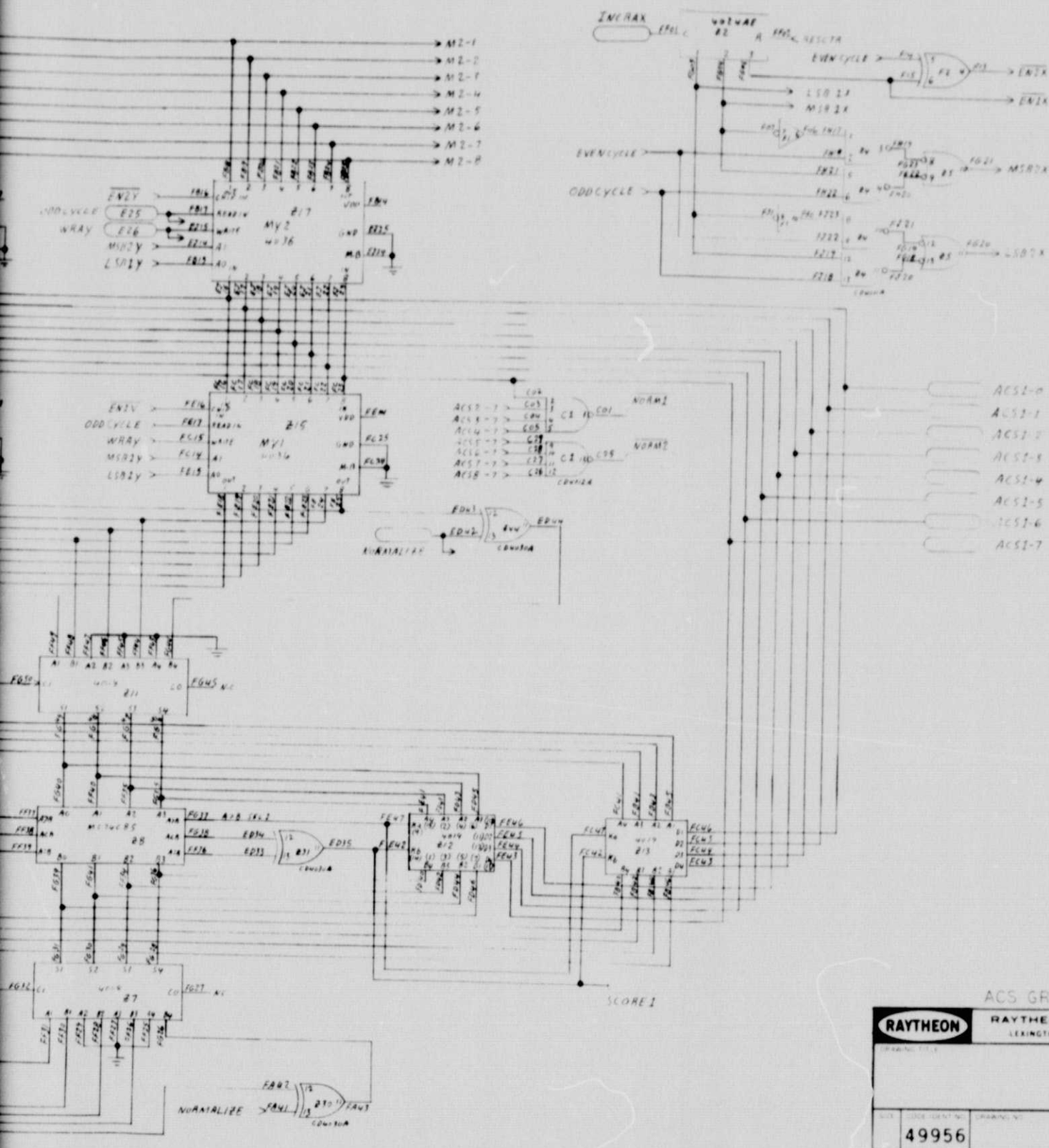


FOLDOUT FRAME 1

ORIGINAL PAGE IS
OF POOR QUALITY

ACS GROUP #1 LOGIC

Figure 2-17



ACS GROUP #1 LOGIC

Figure 2-17

FOLDOUT FRAME 2

The lesser of these new sums is selected and stored in Ay2. The ACS operation for one information bit is completed during the eighth sequence interval where ΔS_4 plus S_4 is compared with $\Delta \bar{S}_4$ plus S_{36} and the smaller sum stored in memory My2, location Ay8.

Normalization of Accumulation Sums

The need for normalization of the accumulated sums is determined by counting the number of "1s" in the most significant bit (MSB) position of all 64 accumulation sums stored during one output bit interval. If all 64 positions are a "1" at the end of the interval, then during the next interval each MSB is complemented to a logic "0" as it is read from memory. Examining the MSB from all eight ACS units, each sequence interval reduces the actual count required to 8. Logic gates C1 on Figure 2-17 generate the two signals $\overline{\text{NORM 1}}$ and $\overline{\text{NORM 2}}$, when all MSB's are a logic "1". This occurrence increments a counter in decoder board #1. At the end of the bit interval (eight sequence intervals) if all 64 MSB's are "1s", the the count will equal 8. In this case a NORMALIZE signal is sent to X-OR gates Z30 and Z40, complementing each MSB logic "1" as it is read from memory during the following bit interval.

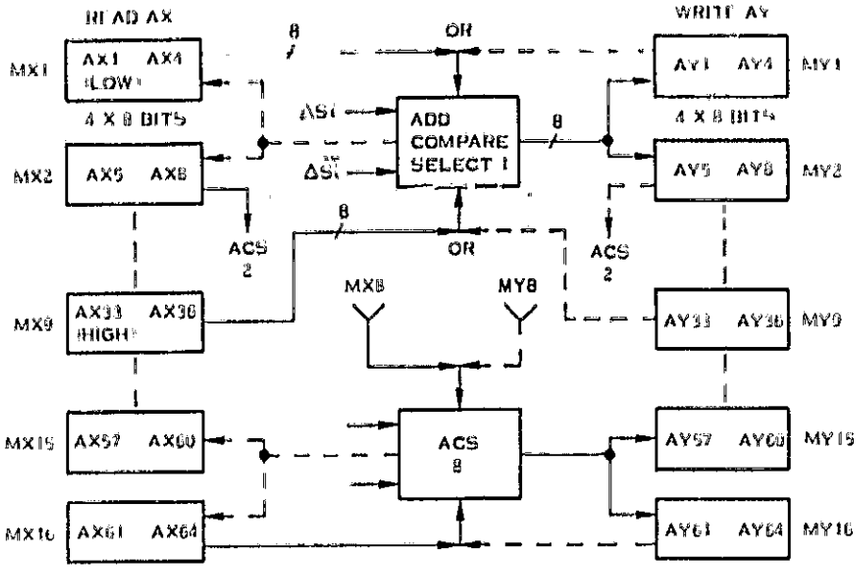
Memory Organization

Figure 2-18 shows in part, the organization of the 32 ACS memory units. The interconnections are shown for the function of reading Ax and writing Ay. A single example of the interconnection for the opposite function is shown with the dashed lines.

The particular memory address organization is shown in Table 2-1. This organization was dictated by the Δ S metric sequence which in turn was selected so that only two Δ S adder circuits are requried. The savings in logic here was achieved with no additional logic in the ACS area.

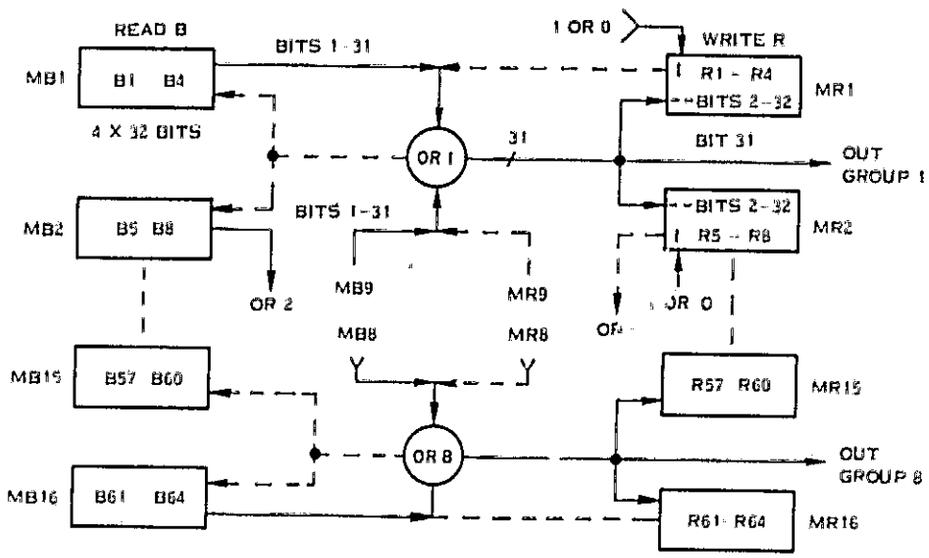
As shown in the upper section of Figure 2-18, Ax locations Ax 1-4 are read from one memory, while locations Ax 33-36 are read from another memory. These two sums are added with the proper Δ S numbers, and the smaller sums stored in Ay locations 1-8. Each memory location on the Ax side is read twice during this interval, producing two new sums which are written into the Ay side.

When the memory functions are reversed in the following bit interval, then Ay locations 4-8 and 33-36 are read at the same time, as shown by the dashed lines, and the smaller ACS result stored in Ax locations 1-8. The "OR" functions between



ACS MEMORY ORGANIZATION

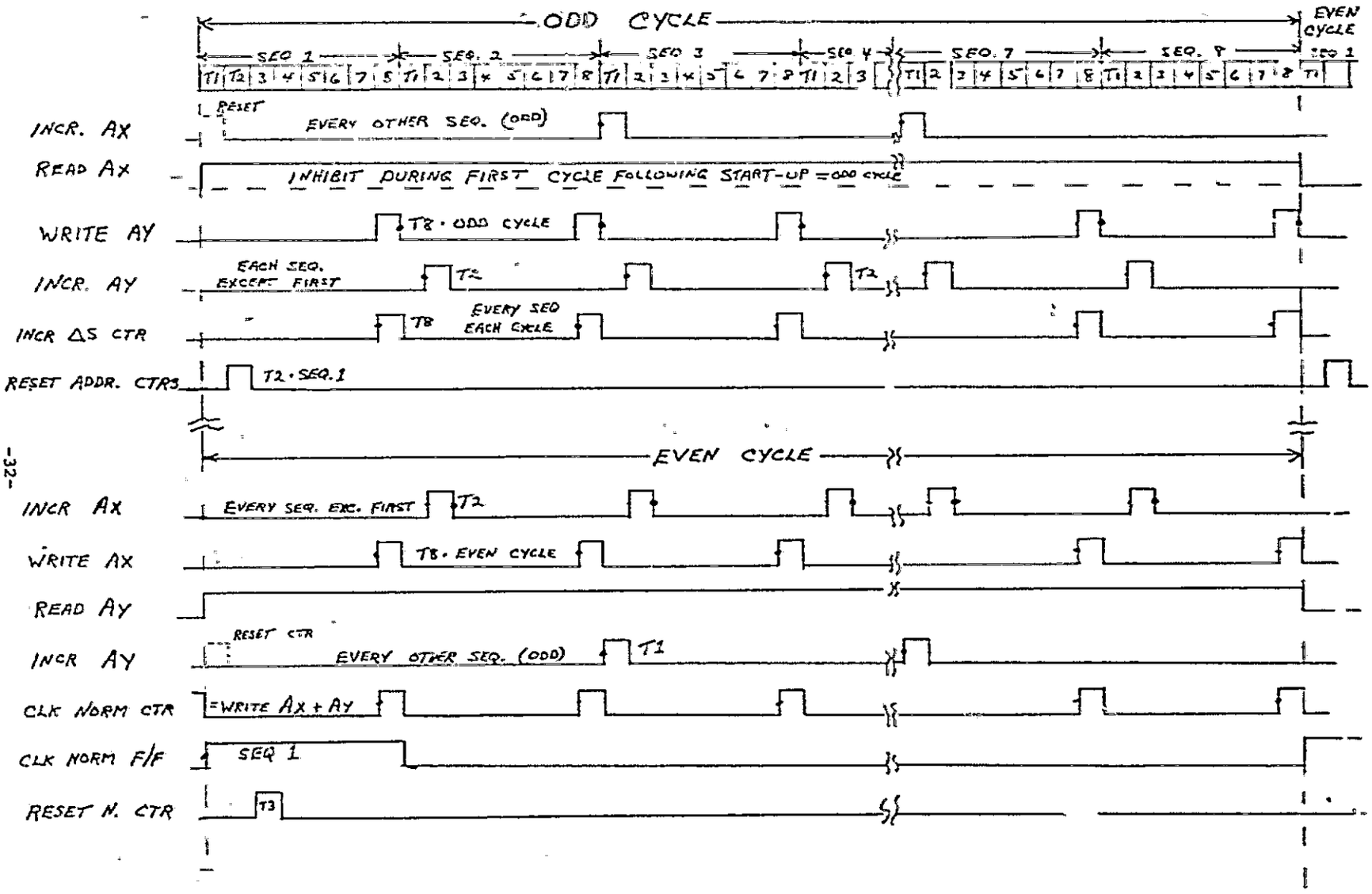
Figure 2-18



PATH MEMORY ORGANIZATION

Figure 2-20

ORIGINAL PAGE IS
OF POOR QUALITY



ACS TIMING

Figure 2-19

Table 2-1

ACS MEMORY MAP

Memory Unit	Physical Location	Memory Contents	Read Sequence	Write Sequence	Input Selected From
1	1	S_1	1,2	1	$S_1+\Delta S_1, S_{33}+\Delta\overline{S_1}$
	2	S_2	3,4	2	$S_1+\Delta\overline{S_1}, S_{33}+\Delta S_1$
	3	S_3	5,6	3	$S_2+\Delta S_2, S_{34}+\Delta\overline{S_2}$
	4	S_4	7,8	4	$S_2+\Delta\overline{S_2}, S_{34}+\Delta S_2$
2	5	S_7	1,2	7	$S_4+\Delta S_4, S_{36}+\Delta\overline{S_4}$
	6	S_8	3,4	8	$S_4+\Delta\overline{S_4}, S_{36}+\Delta S_4$
	7	S_5	5,6	5	$S_3+\Delta S_3, S_{35}+\Delta\overline{S_3}$
	8	S_6	7,8	6	$S_3+\Delta\overline{S_3}, S_{35}+\Delta S_3$
3	9	S_{11}	1,2	7	$S_6+\Delta S_4, S_{42}+\Delta\overline{S_4}$
	10	S_{12}	3,4	8	$S_6+\Delta\overline{S_4}, S_{42}+\Delta S_4$
	11	S_9	5,6	5	$S_5+\Delta S_3, S_{37}+\Delta\overline{S_3}$
	12	S_{10}	7,8	6	$S_5+\Delta\overline{S_3}, S_{37}+\Delta S_3$
4	13	S_{13}	1,2	1	$S_7+\Delta S_1, S_{39}+\Delta\overline{S_1}$
	14	S_{14}	3,4	2	$S_7+\Delta\overline{S_1}, S_{39}+\Delta S_1$
	15	S_{15}	5,6	3	$S_8+\Delta S_2, S_{40}+\Delta\overline{S_2}$
	16	S_{16}	7,8	4	$S_8+\Delta\overline{S_2}, S_{40}+\Delta S_2$
5	17	S_{18}	1,2	5	$S_9+\Delta S_3, S_{41}+\Delta\overline{S_3}$
	18	S_{17}	3,4	6	$S_9+\Delta\overline{S_3}, S_{41}+\Delta S_3$
	19	S_{20}	5,6	7	$S_{10}+\Delta S_4, S_{42}+\Delta\overline{S_4}$
	20	S_{19}	7,8	8	$S_{10}+\Delta\overline{S_4}, S_{42}+\Delta S_4$
6	21	S_{24}	1,2	3	$S_{12}+\Delta S_2, S_{44}+\Delta\overline{S_2}$
	22	S_{23}	3,4	4	$S_{12}+\Delta\overline{S_2}, S_{44}+\Delta S_2$
	23	S_{22}	5,6	1	$S_{11}+\Delta S_1, S_{43}+\Delta\overline{S_1}$
	24	S_{21}	7,8	2	$S_{11}+\Delta\overline{S_1}, S_{43}+\Delta S_1$
7	25	S_{25}	1,2	2	$S_{13}+\Delta\overline{S_1}, S_{45}+\Delta S_1$
	26	S_{26}	3,4	1	$S_{13}+\Delta S_1, S_{45}+\Delta\overline{S_1}$
	29	S_{27}	5,6	4	$S_{14}+\Delta\overline{S_2}, S_{46}+\Delta S_2$
	28	S_{28}	7,8	3	$S_{14}+\Delta S_2, S_{46}+\Delta\overline{S_2}$
8	29	S_{31}	1,2	8	$S_{16}+\Delta\overline{S_4}, S_{48}+\Delta S_4$
	30	S_{32}	3,4	7	$S_{16}+\Delta S_4, S_{48}+\Delta\overline{S_4}$
	31	S_{29}	5,6	6	$S_{15}+\Delta\overline{S_3}, S_{47}+\Delta S_3$
	32	S_{30}	7,8	5	$S_{15}+\Delta S_3, S_{47}+\Delta\overline{S_3}$

Table 2-1
(Continued)

Memory Unit	Physical Location	Memory Contents	Read Sequence	Write Sequence	Input Selected From
9	33	S_{33}	1,2	4	$S_{17}+\Delta\overline{S_2}, S_{49}+\Delta S_2$
	34	S_{34}	3,4	3	$S_{17}+\Delta S_2, S_{49}+\Delta\overline{S_2}$
	35	S_{35}	5,6	2	$S_{18}+\Delta\overline{S_1}, S_{50}+\Delta S_1$
	36	S_{36}	7,8	1	$S_{18}+\Delta S_1, S_{50}+\Delta\overline{S_1}$
10	37	S_{38}	1,2	6	$S_{20}+\Delta\overline{S_3}, S_{52}+\Delta S_3$
	38	S_{40}	3,4	5	$S_{20}+\Delta S_3, S_{52}+\Delta\overline{S_3}$
	39	S_{37}	5,6	8	$S_{19}+\Delta\overline{S_4}, S_{51}+\Delta S_4$
	40	S_{38}	7,8	7	$S_{19}+\Delta S_4, S_{51}+\Delta\overline{S_4}$
11	41	S_{43}	1,2	6	$S_{22}+\Delta\overline{S_3}, S_{54}+\Delta S_3$
	42	S_{44}	3,4	5	$S_{22}+\Delta S_3, S_{54}+\Delta\overline{S_3}$
	43	S_{41}	5,6	8	$S_{21}+\Delta\overline{S_4}, S_{53}+\Delta S_4$
	44	S_{42}	7,8	7	$S_{21}+\Delta S_4, S_{53}+\Delta\overline{S_4}$
12	45	S_{45}	1,2	4	$S_{23}+\Delta\overline{S_2}, S_{55}+\Delta S_2$
	46	S_{46}	3,4	3	$S_{23}+\Delta S_2, S_{55}+\Delta\overline{S_2}$
	47	S_{46}	5,6	2	$S_{24}+\Delta\overline{S_1}, S_{56}+\Delta S_1$
	48	S_{48}	7,8	1	$S_{24}+\Delta S_1, S_{56}+\Delta\overline{S_1}$
13	49	S_{50}	1,2	1	$S_{25}+\Delta S_1, S_{57}+\Delta\overline{S_1}$
	50	S_{49}	3,4	2	$S_{25}+\Delta\overline{S_1}, S_{57}+\Delta S_1$
	51	S_{52}	5,6	3	$S_{26}+\Delta S_2, S_{58}+\Delta\overline{S_2}$
	52	S_{51}	9,8	4	$S_{26}+\Delta\overline{S_2}, S_{58}+\Delta S_2$
14	53	S_{56}	1,2	7	$S_{28}+\Delta S_4, S_{60}+\Delta\overline{S_4}$
	54	S_{55}	3,4	8	$S_{28}+\Delta\overline{S_4}, S_{60}+\Delta S_4$
	55	S_{54}	5,6	5	$S_{27}+\Delta S_3, S_{59}+\Delta\overline{S_3}$
	56	S_{53}	7,8	6	$S_{27}+\Delta\overline{S_3}, S_{59}+\Delta S_3$
15	57	S_{57}	1,2	6	$S_{29}+\Delta\overline{S_3}, S_{61}+\Delta S_3$
	58	S_{58}	3,4	5	$S_{29}+\Delta S_3, S_{61}+\Delta\overline{S_3}$
	59	S_{59}	5,6	8	$S_{30}+\Delta\overline{S_4}, S_{62}+\Delta S_4$
	60	S_{60}	7,8	7	$S_{30}+\Delta S_4, S_{62}+\Delta\overline{S_4}$
16	61	S_{63}	1,2	4	$S_{32}+\Delta\overline{S_2}, S_{64}+\Delta S_2$
	62	S_{64}	3,4	3	$S_{32}+\Delta S_2, S_{64}+\Delta\overline{S_2}$
	63	S_{61}	5,6	2	$S_{31}+\Delta\overline{S_1}, S_{63}+\Delta S_1$
	64	S_{62}	7,8	1	$S_{31}+\Delta S_1, S_{63}+\Delta\overline{S_1}$

Ax and Ay memories is accomplished through "wire-oring" of the memory outputs.

Memory Addressing

On the lower left side of Figure 2-18 is the MY memory address and enable logic for the entire ACS board. The corresponding logic for the MX memories is on the upper right side of Figure 2-18.

All memory addresses (cf. Table 2-1) are read sequentially during the read operation. In order to allow this sequential addressing while reading, certain units, M2, M4, M8, M9 and M11, must be addressed, when writing, in a 3, 4, 1, 2 sequence; M13 and M6 are addressed in a 2, 1, 4, 3 sequence: M5 and M14 in a 4, 3, 2, 1 sequence; and the remainder are addressed in the normal sequential manner.

The memory enable signals shown in Figure 2-18 are used as part of the address control during write operation. When the address count is below 4, the memory locations 1-4 are enabled, while locations 5-8 are enabled when the count is 4 or greater.

ACS Timing

The timing diagram for the ACS section is shown in Figure 2-19. All time periods are based on a maximum information bit rate of 100 K bits per second. For lower bit rates, the decoder operates at the same rate, then halts in an idle state until the next three input signals are received.

The 10 μsec bit period is divided into eight 1.25 μsec sequence intervals. Each sequence interval is then further divided into eight time pulses, T1-T8, of 156 nsec duration.

The memory address is changed at the start of the T1 time and the read output allowed to ripple through the adder/comparator combination. No data set-up time is required for the memory units; a maximum hold time of 40 nsec is sufficient. The CMOS memory unit is the 4 word by 8 bit, RCA 4036.

2.2.3 Path Memory Logic

This section consists of the path memory and the associated output bit selection logic. The path memory (reference Figure 2-20) is implemented with 128 CD4036 memory units. These units are connected to provide dual 64 word by 32 bit memory sections, B1-B46, and R1-R46. The 64 words are then subdivided into eight 8-word functional groups. The ACS comparison result from group number one is used to determine whether to read from B1-B4 in MB1, or from B33-B36 in MB9. Data from

the selected memory, bits 1-31 are then written into location R1 of MR1, bits 2-32. A 1 or 0 is written into bit 1 = PM1 AUG., depending on the result of the comparison. Detailed logic diagrams for the path memory are on Drawings VDBD #3, #4 and #5.

Path Memory Addressing

The path memory addressing organization is the same as that for the ACS memory described in section 2.2.2. Address and enable logic is shown in Figure 2-21.

Memory Read Control

Logic for controlling the read signals to each memory is shown on Figure 2-22. The eight score signals from the ACS sections are level converted from +10 VDC logic to +8 VDC logic with buffers E14 and E15, then trapped in flip-flops E9 through E12.

When the Read MB signal is received during the odd cycle, it is directed to one memory in each group, according to the trapped scores. Identical selection gates are provided for MR memory read control during even cycles.

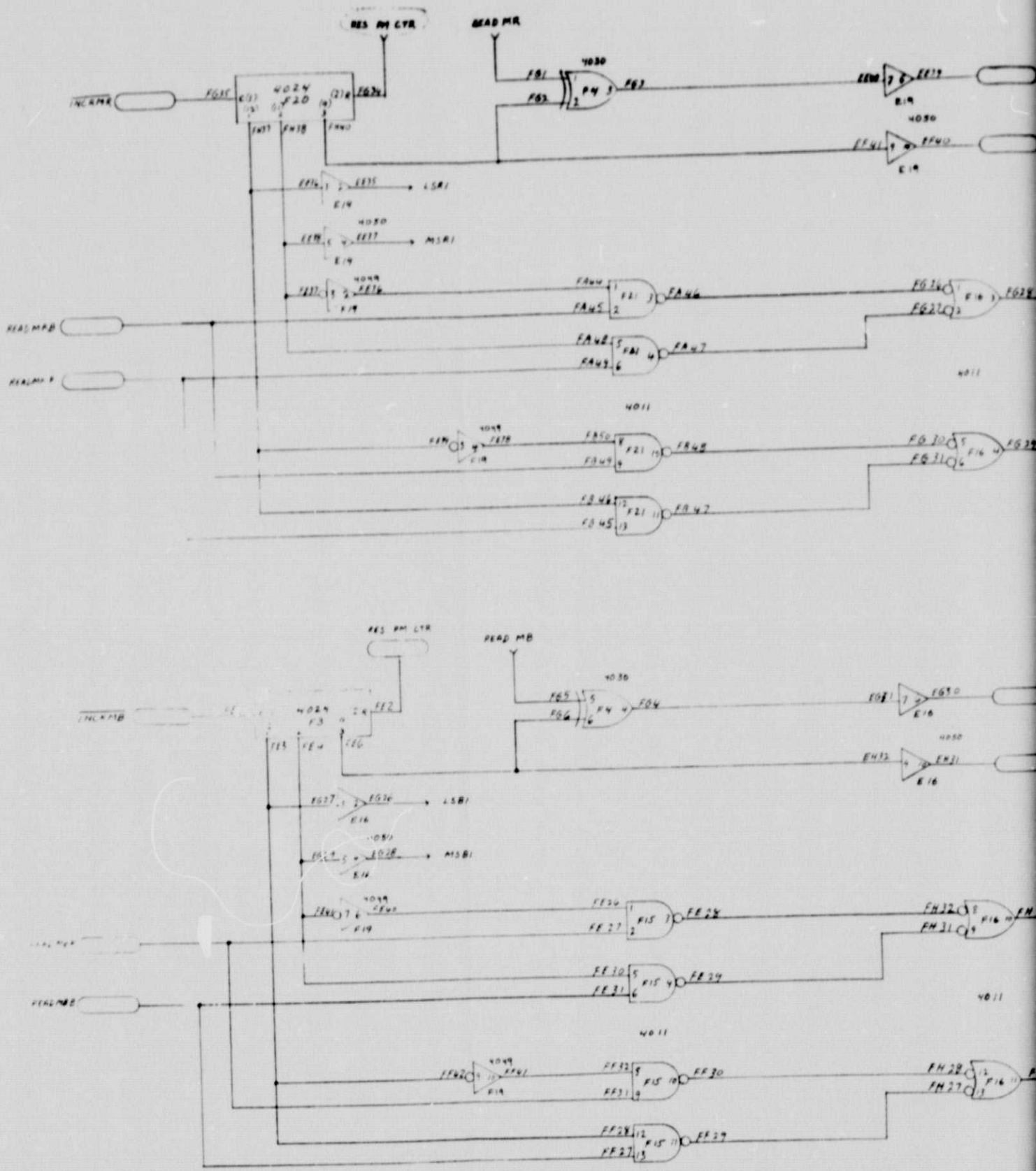
Typical Path Memory Functional Group

The interconnections for one-half of a typical path memory group, in this example MB1 and MB2, are shown on Figure 2-23. The remaining half of the group consists of MR1 and MR2. This sheet serves as a master for all sixteen similar sections. Control signals and PM input and output numbers are assigned to comply with each location.

Output Bit Selection

Logic for output bit selection is shown in Figure 2-24. The output bit is selected by counting the number of logic "1s" in the oldest bit position (bit 32) of all 64 memory locations. If the count is less than or equal to 32, then the output is a logic 0; if the count is greater than 32, the output is a logic "1".

One 5-stage counter is multiplexed between two path memory groups, requiring a total of four counters. A memory in each group is loaded once per sequence interval. The write input for bit position 32 is brought out for each group (labeled MSB on the drawing) and connected to the and-or-select gates. Data being written into memory is held long enough to be strobed into the counter by enable signals EN1 and EN2.

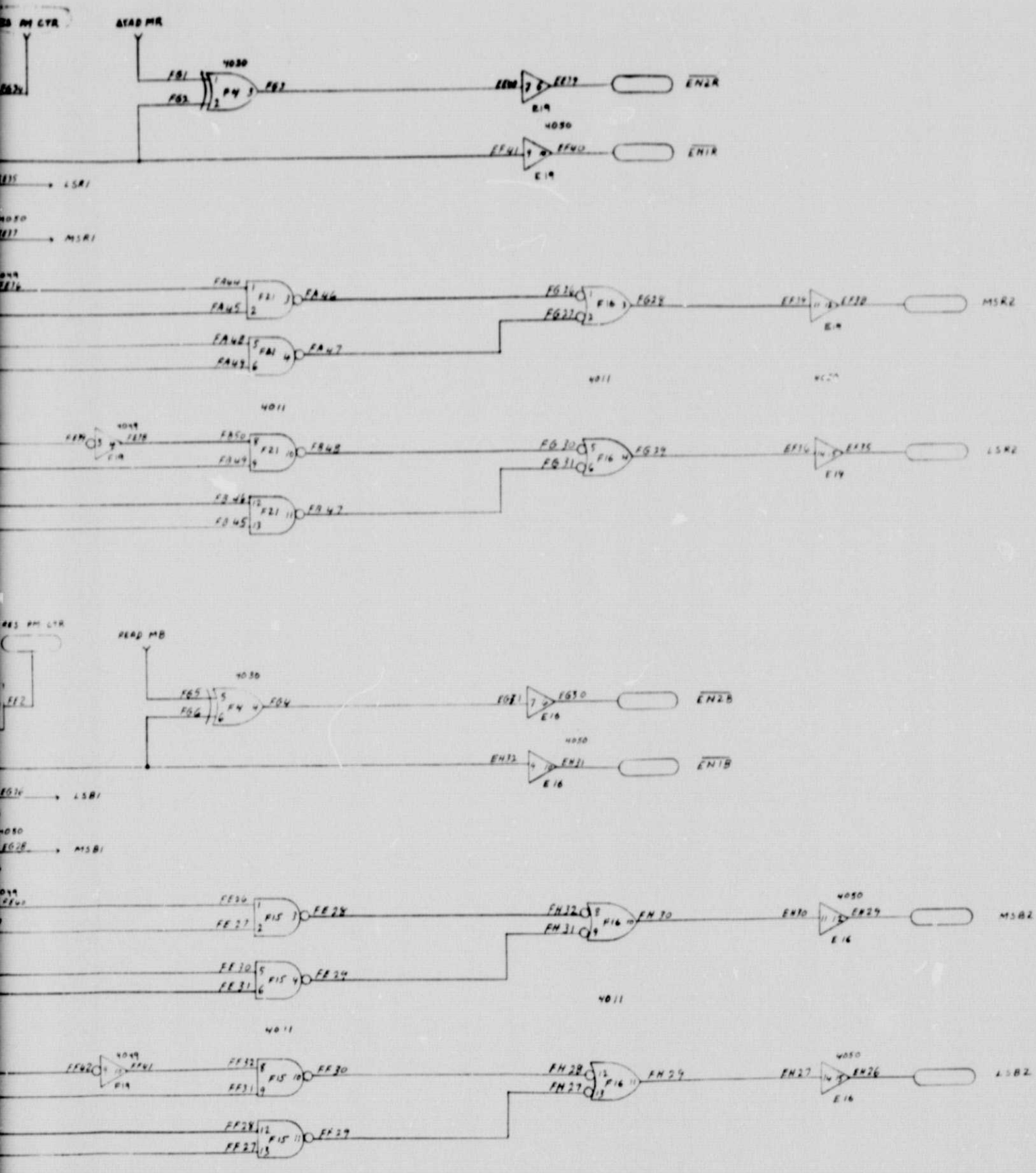


PATH MEMORY ADDRESS & ENABLE LOGIC

Figure 2-21

FOLDOUT FRAME /

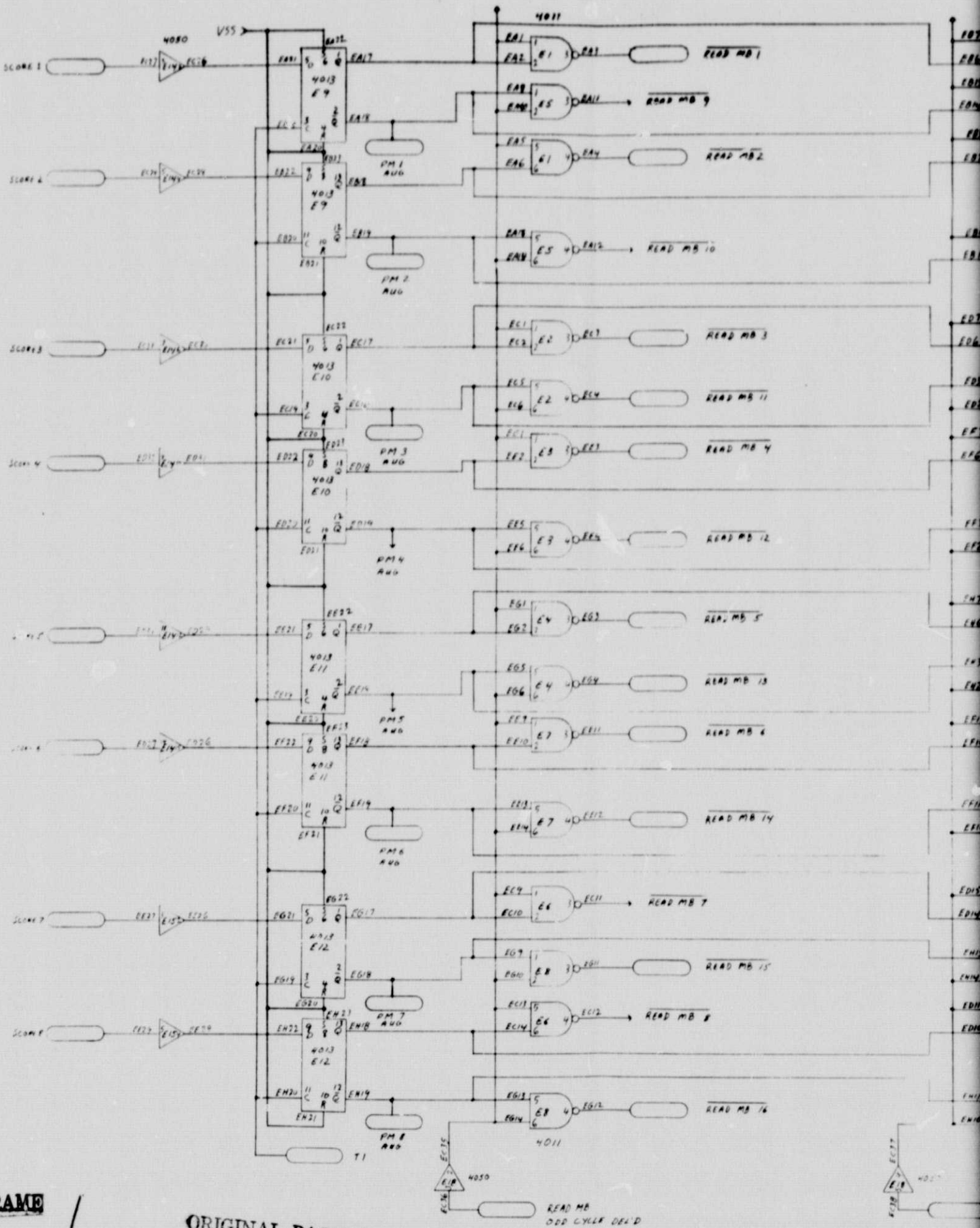
ORIGINAL PAGE IS
OF POOR QUALITY



PATH MEMORY ADDRESS & ENABLE LOGIC

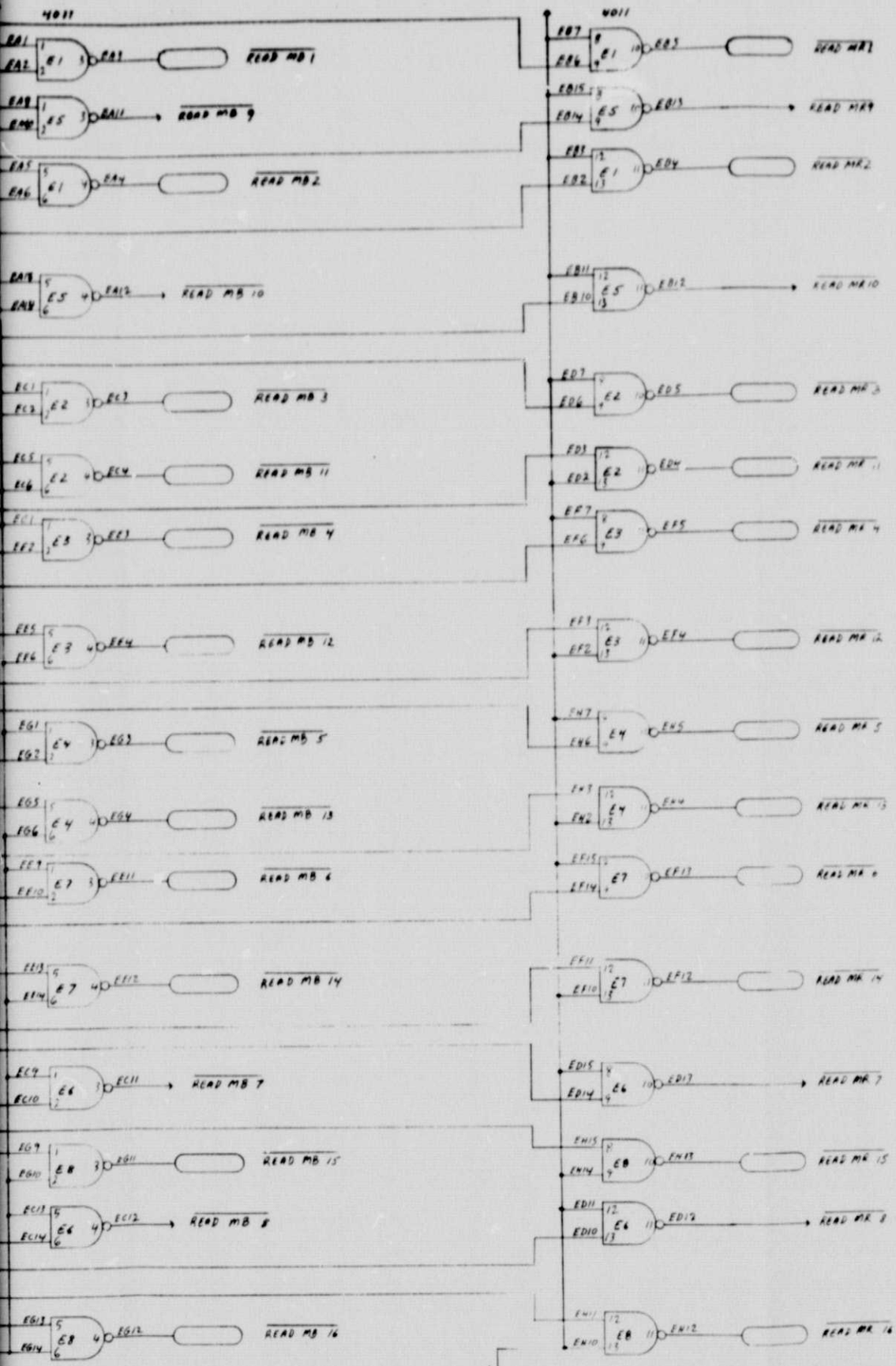
Figure 2-21

FOLDOUT FRAME



FOLDOUT FRAME

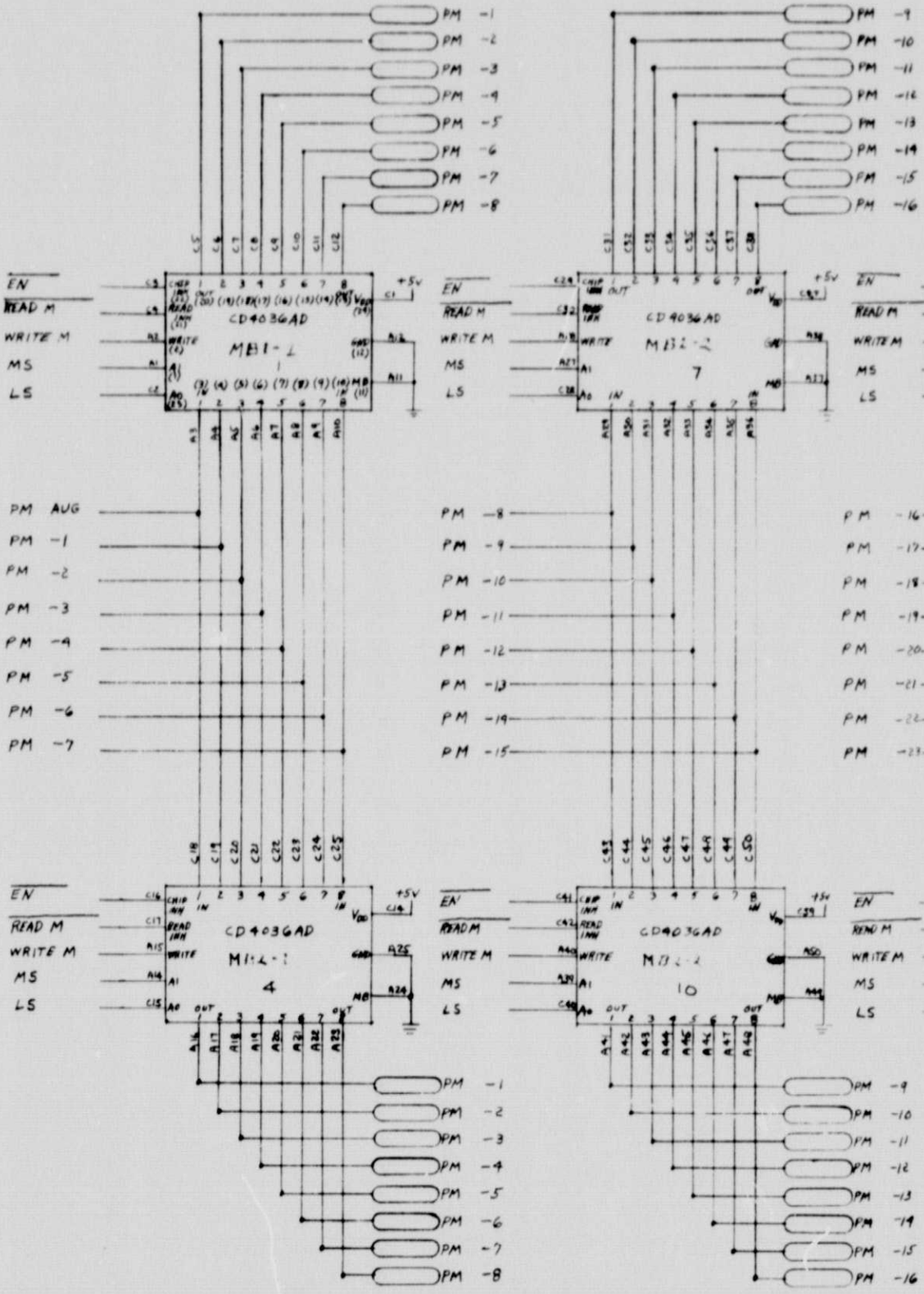
ORIGINAL PAGE IS
OF POOR QUALITY



PATH MEMORY READ CONTROL LOGIC

Figure 2-22

FOLDOUT FRAME

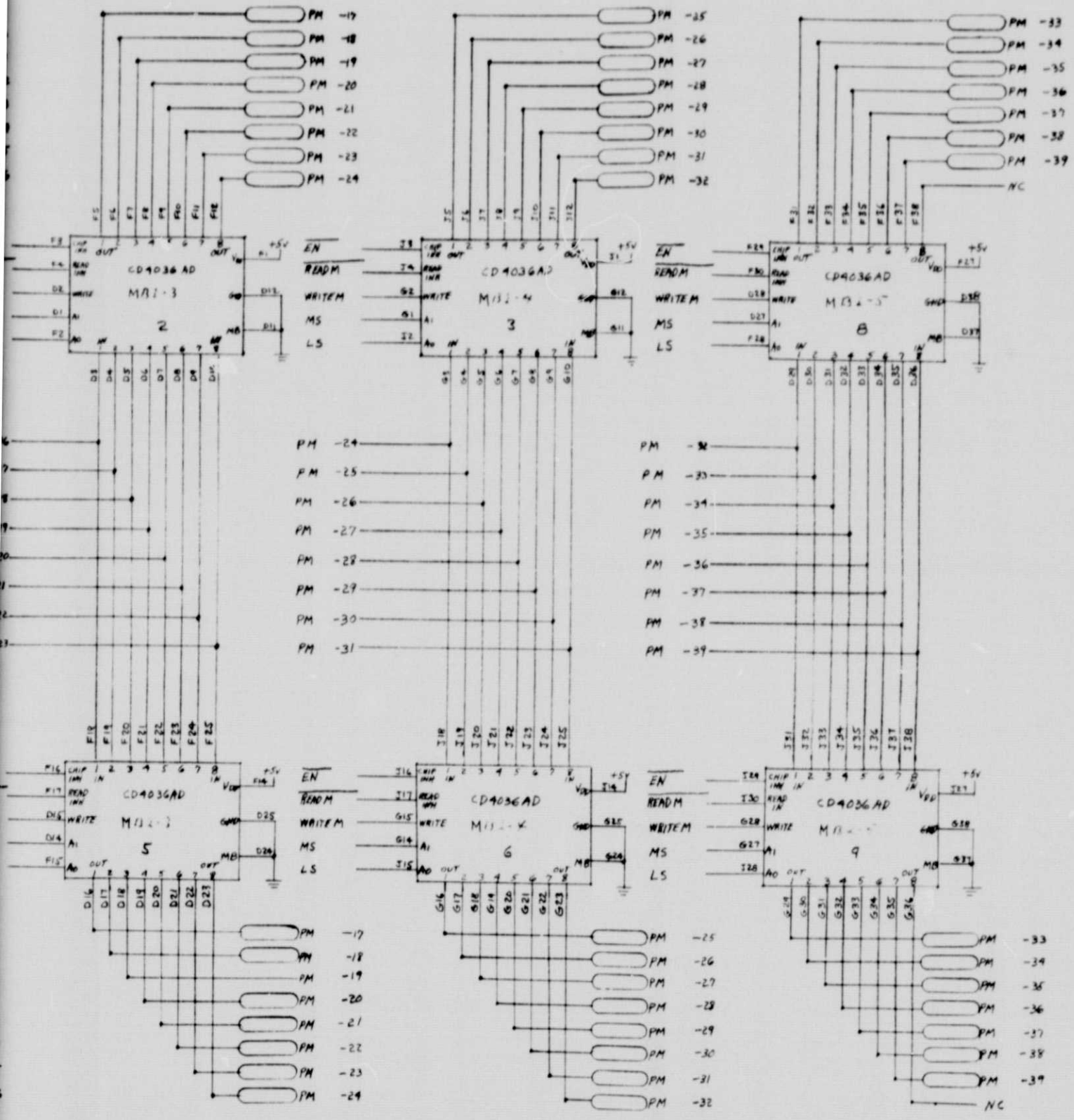


TYPICAL LOGIC FOR
A PATH MEMO

Figure 2

FOLDOUT FRAME /

ORIGINAL PAGE IS
OF POOR QUALITY

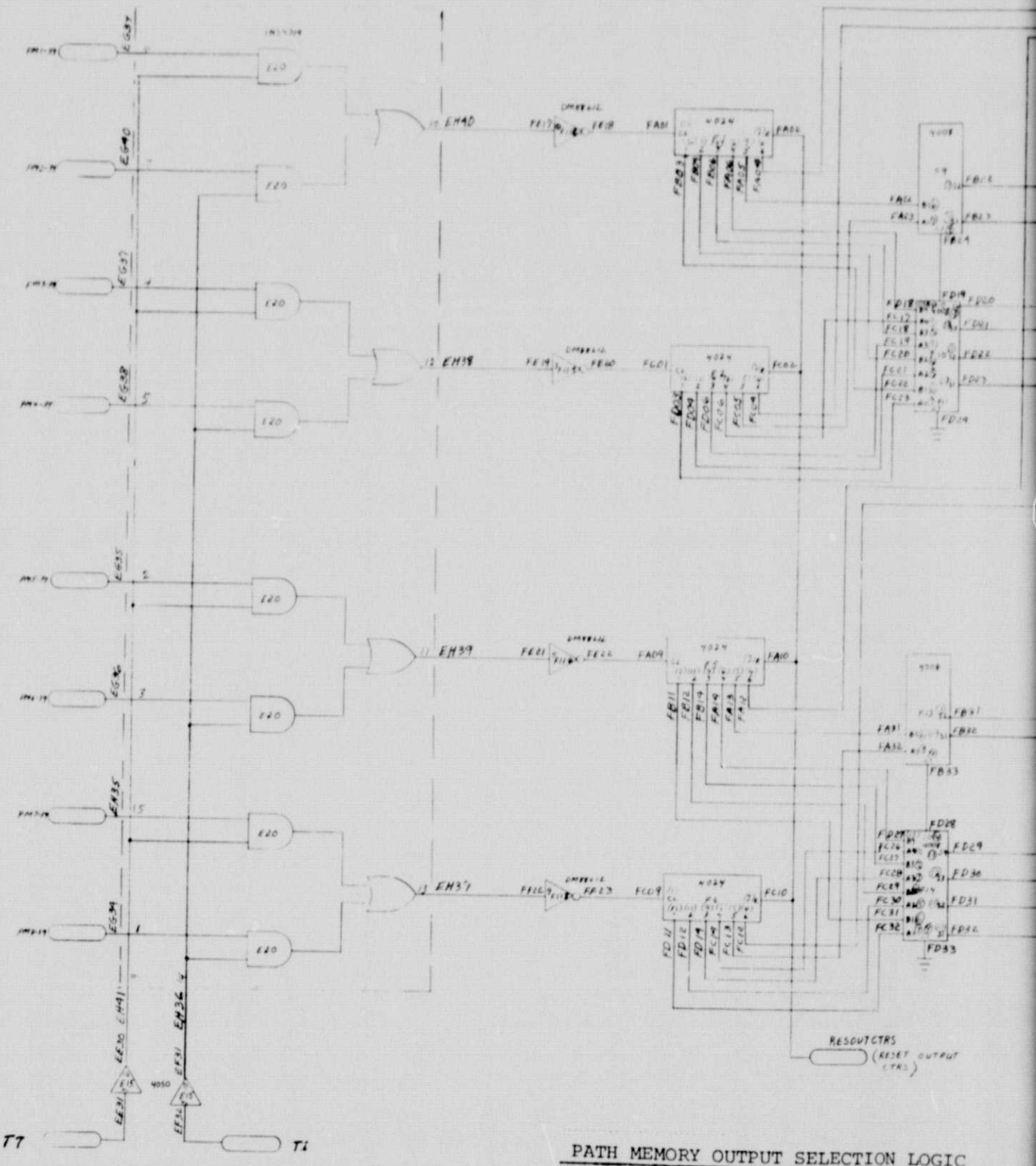


TYPICAL LOGIC FOR ONE-HALF OF A PATH MEMORY GROUP

Figure 2-23

PATH MEMORY

FOLDOUT FRAME 2

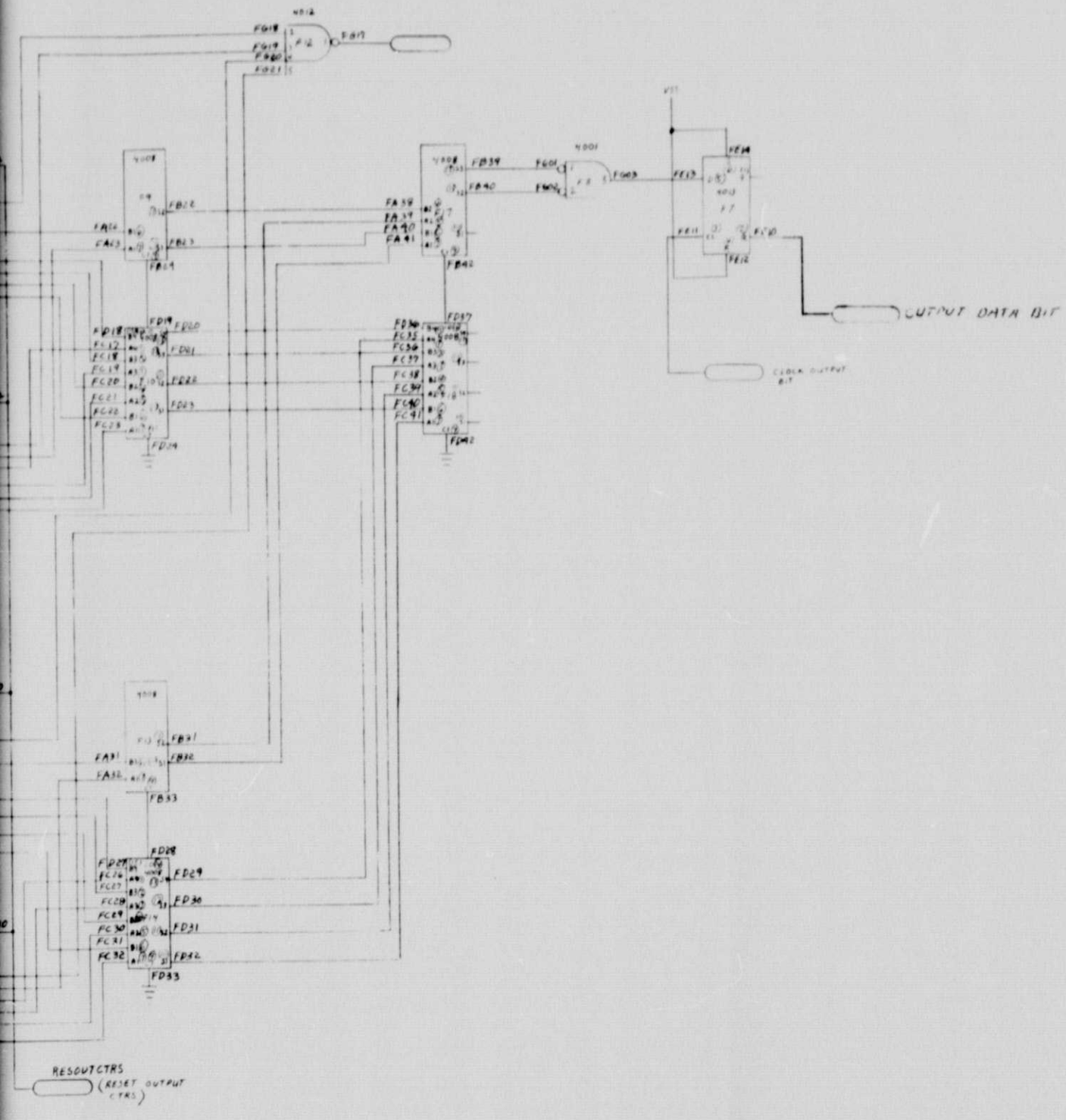


PATH MEMORY OUTPUT SELECTION LOGIC

Figure 2-24

FOLDOUT FRAME /

ORIGINAL PAGE IS
OF POOR QUALITY



T SELECTION LOGIC

e 2-24

POOR QUALITY

0-

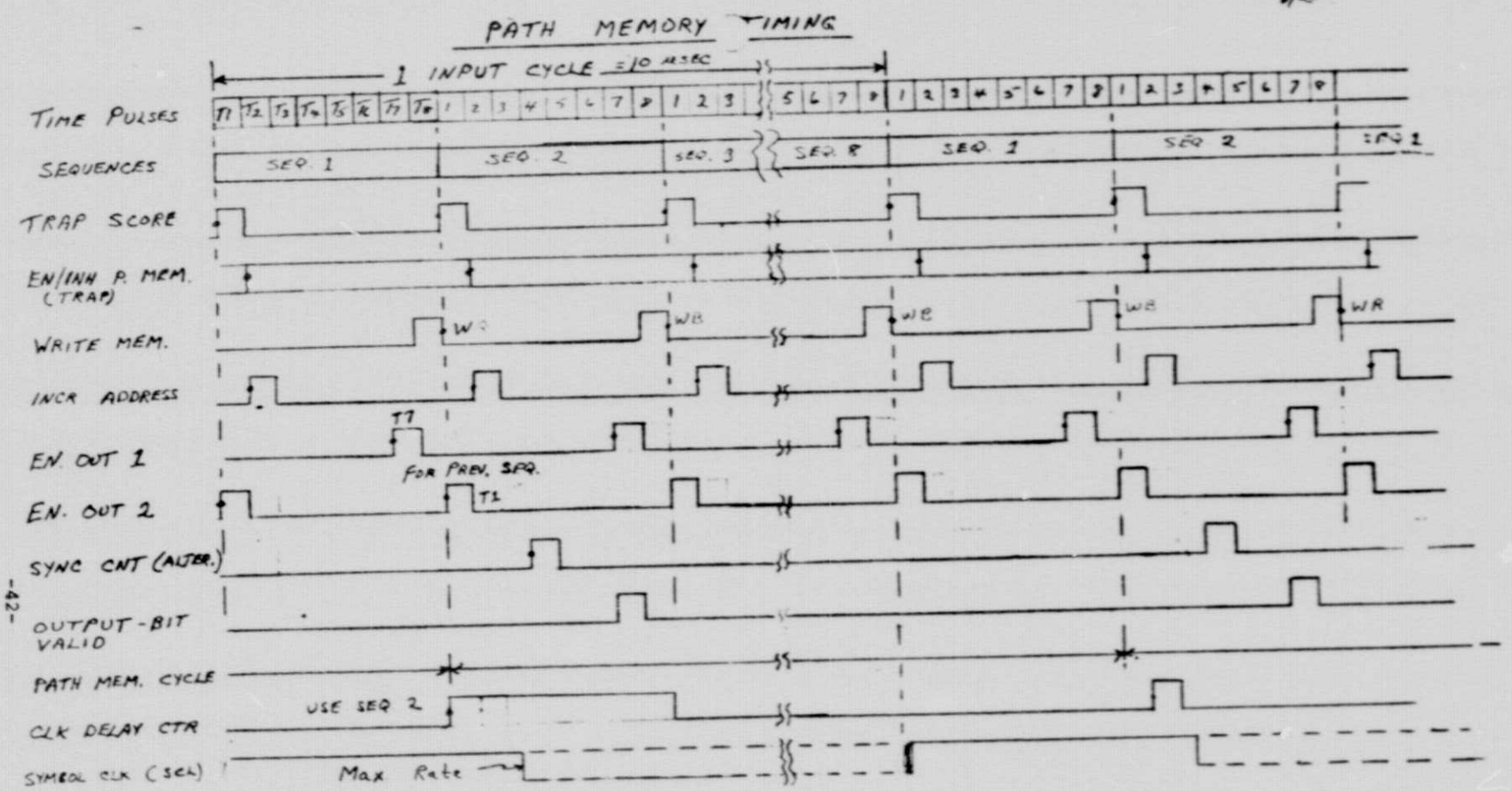
Following the eighth sequence interval which is the end of the bit interval, the contents of the counters are added and compared with binary 32 for the output indication. This indication is clocked into a flip-flop and sent to VDBD #1, Sheet 1, where it is level converted to a 9614 TTL output driver. The counter inputs are level converted from 8 to 10 volts and the output counters, adders and comparators are operated at 10 VDC.

Path Memory Timing

Basic timing for the path memory is shown in Figure 2-25. The output bit is available at the end of sequence 2 of the following bit interval.

Path Memory Packaging

Path memory logic is packaged on three Augat 54-row universal packaging panels. Each of two panels holds 48 memories, while the remaining 32 memories are mounted on the third panel, along with the addressing and output control logic.



PATH MEMORY TIMING

Figure 2-25

III. TEST SET DESIGN AND IMPLEMENTATION

The major function of the test set is to provide noise modulated data to the Viterbi Decoder. It perfoms this function by encoding random data (changing at a variable rate of up to 100 KBPS) with a rate 1/3 convolutional encoder. It modifies this data with the output of a random noise generator and transfers the resultant data as three parallel bits at three times the original data frequency. This data is sent to the Viterbi Decoder along with a clock. Data returned from the Viterbi Decoder is compared with the original data and the number of errors generated can be recorded by an external counter. Figure 3-1 is a general block diagram of the test set with each block containing the figure number of three more detailed block diagrams. Each block in Figures 3-1, 3-2, and 3-3 contains the sheet number of the logic diagram wherein it originates.

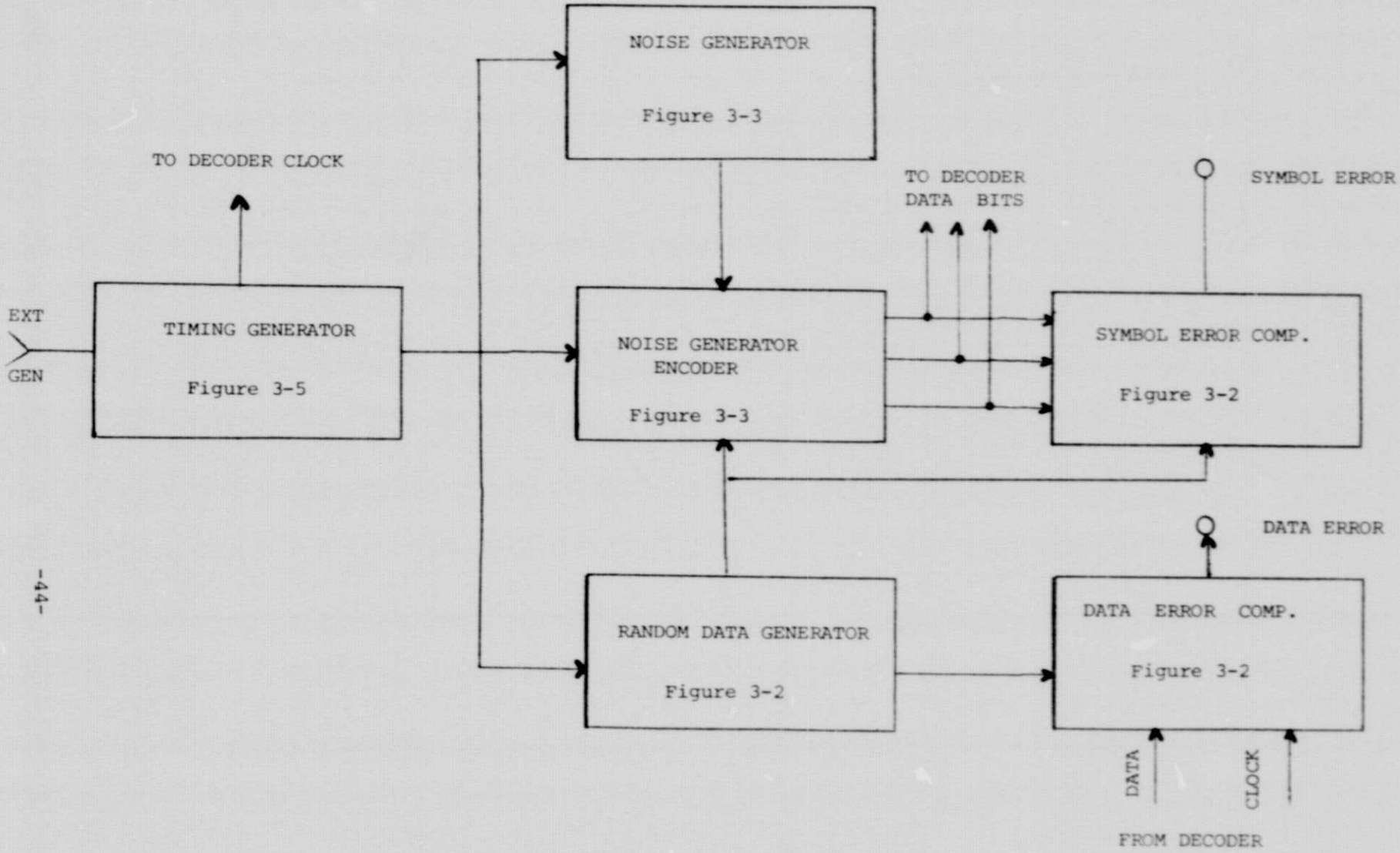
The Random Data Generator (Figure 3-2) consists of a 24-stage SR (shift register) wirh appropriate exclusive OR circuits feeding back various outputs into the first stage. A data SR of up to 48 stages is provided to accommodate the delay of the data in moving through the path memory in the Viterbi Decoder to insure that the returned data will be in phase with the original data at the data error comparator. The decoder clock from the Viterbi Decoder will assure that the two data signals are compared at the proper time. The convolutional encoder input consists of seven lines from the random data generator. The resultant data is sequentially presented on the encoder output at three times the data generator rate.

The Noise Encoder (Figure 3-3) combines the output of the noise comparators with the convolutional encoder output and the resultant three outputs are stored in D flip-flops. The outputs of these flip-flops are Data BITS 1, 2, and 3 for the Viterbi Decoder.

The Symbol Error Comparator (Figure 3-2) compares the encoder output with the three data bits. The symbol errors generated by the noise can be counted with an external counter.

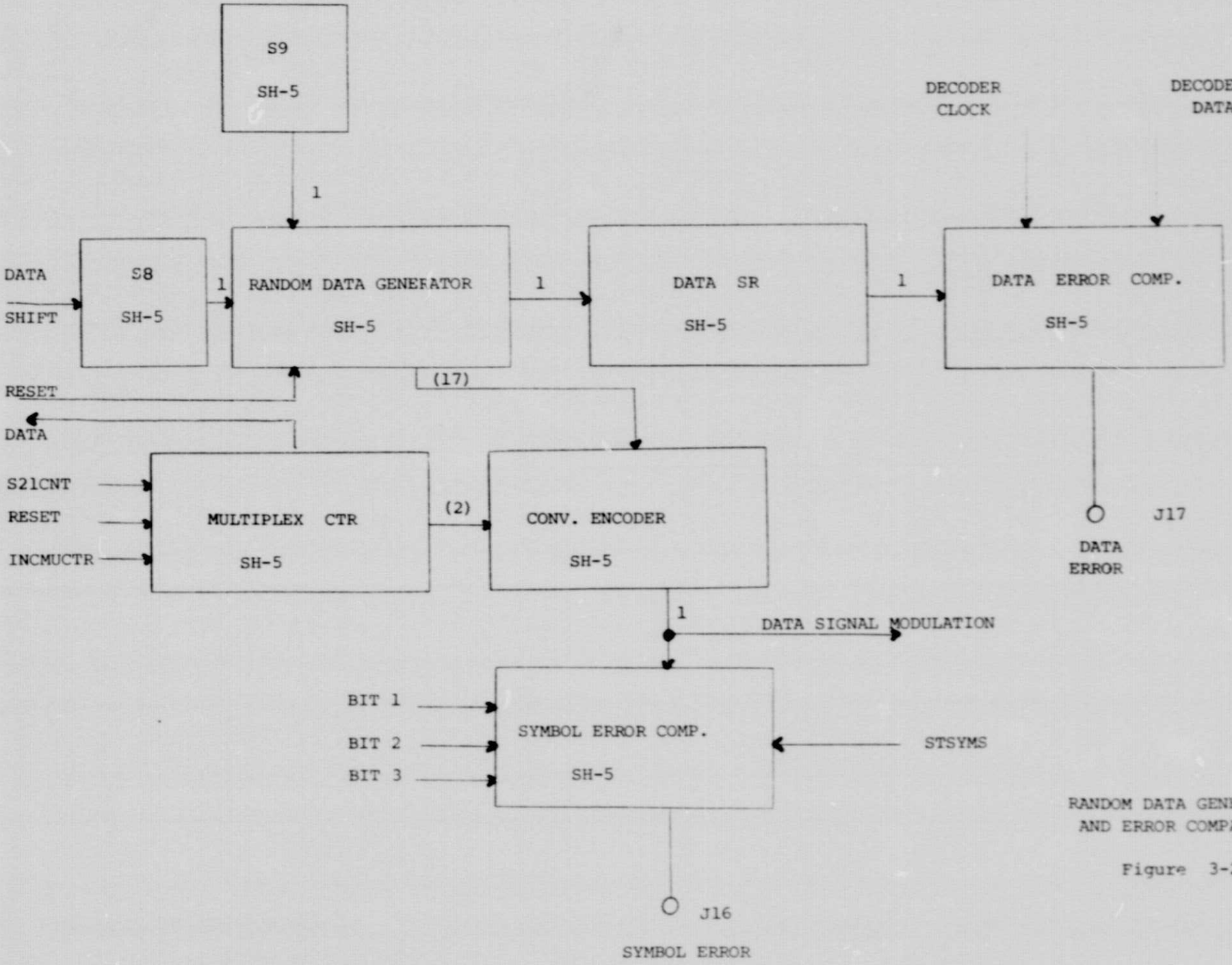
The Noise Generator (Figure 3-3) consists of 6 ROM memories (256 x 8); 5 twenty-one bit D-registers; a random noise generator; 7 twenty-one bit comparators; a noise generator encoder; a two-stage counter and a priority encoder.

The original noise source used one 63-stage shift-register rather than the 19-, 20-, and 23-stage shift-registers now being used. The 63-stage shift-register had several deficiencies that were substantially eliminated by changing to the shorter registers. The periods of the overall sequences generated by the two



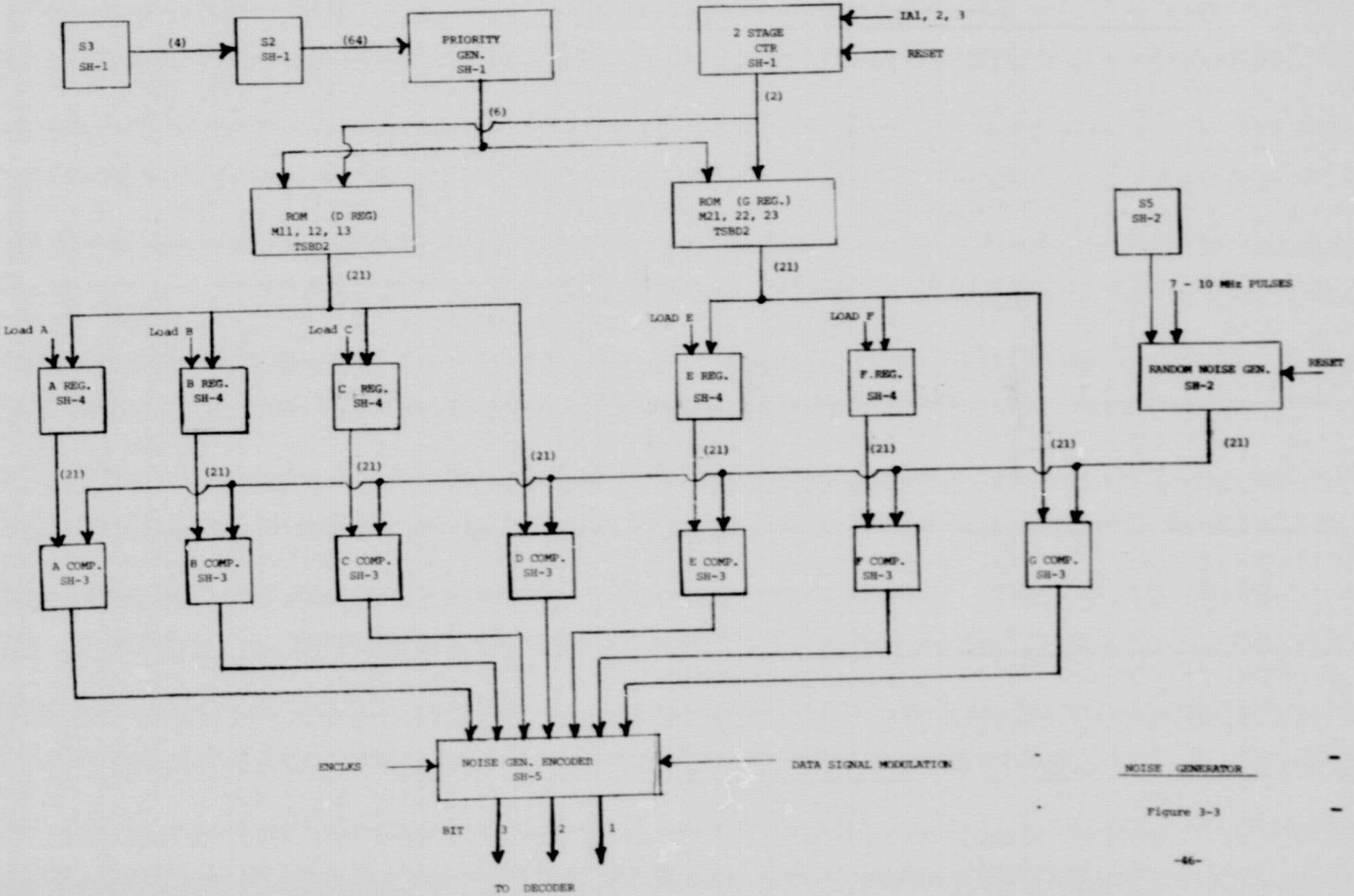
GENERAL BLOCK DIAGRAM

Figure 3-1



RANDOM DATA GENERATOR AND ERROR COMPARATORS

Figure 3-2



NOISE GENERATOR

Figure 3-3

methods are of the same order of magnitude (roughly 10^4 years and 3×10^4 years, respectively). The 63-stage register, however, takes the full 10,000 years to go through all of its states whereas each of the shorter registers goes through all of its states in roughly 1/2 second. The long register thus cycles through only an infinitesimal fraction of its possible states during any one run while each of the short registers completes a number of cycles. As a consequence, the long register can spend an entire run in a relatively atypical portion of its sequence. This defect was compounded by the fact that trinominal feedback was used to generate the sequence. If the register cells were rich in zeros at the beginning of a run (which they tended to be even without the "initialize" switch in the ON position) they would stay rich in zeros for some time. This resulted in an atypically bursty noise environment and biased the results accordingly. The effect was negligible at low signal-to-noise ratios but introduced a degradation of roughly 1/2 dB at bit error rates of 10^{-16}. Although the two generators results in identical first-order statistics, the second approach was felt to be more representative of the anticipated noise in that it is able to cycle through a full range of "typical" states during any one run.

The desired noise and threshold levels are obtained by setting the threshold switch (S3) and the S/N dB switch S2. Sixty-four lines from S2 are inputs to a priority encoder whose output generates the six least significant bits of the ROM address. The two-stage counter determines the two most significant bits of the ROM address. Whenever the start button is depressed a sequence of pulses IA1, IA2, and IA3 increment the two-stage counter. The contents of the ROM's are stored in registers A, B, C, E and F as indicated in Figure 3-6. A burst of twenty-one 10 MHz pulses shift the data in the 63-bit noise shift register. The twenty-one output lines from this noise register are compared with the noise data from registers A, B, C, D, E, F, and G. The 7 comparator outputs (NA, NB,.....,NG) are inputs to the noise generator encoder.

The Timing Generator (Figure 3-5) consists of a 10 MHz oscillator; 8-stage Johnson Counter; 6-stage binary counter; a 7-counter burst generator timing logic and control logic. The HALT button resets all counters and presets the noise generator shift register and data generator shift register to anone in the first stage and zeros in the remaining stages if the corresponding initialize switch S5 (Figure 2) and S9 (Figure 1) are in the up position. If S5 and S9 are down, the HALT switch has no effect; thus the register contents remain random. Depressing the

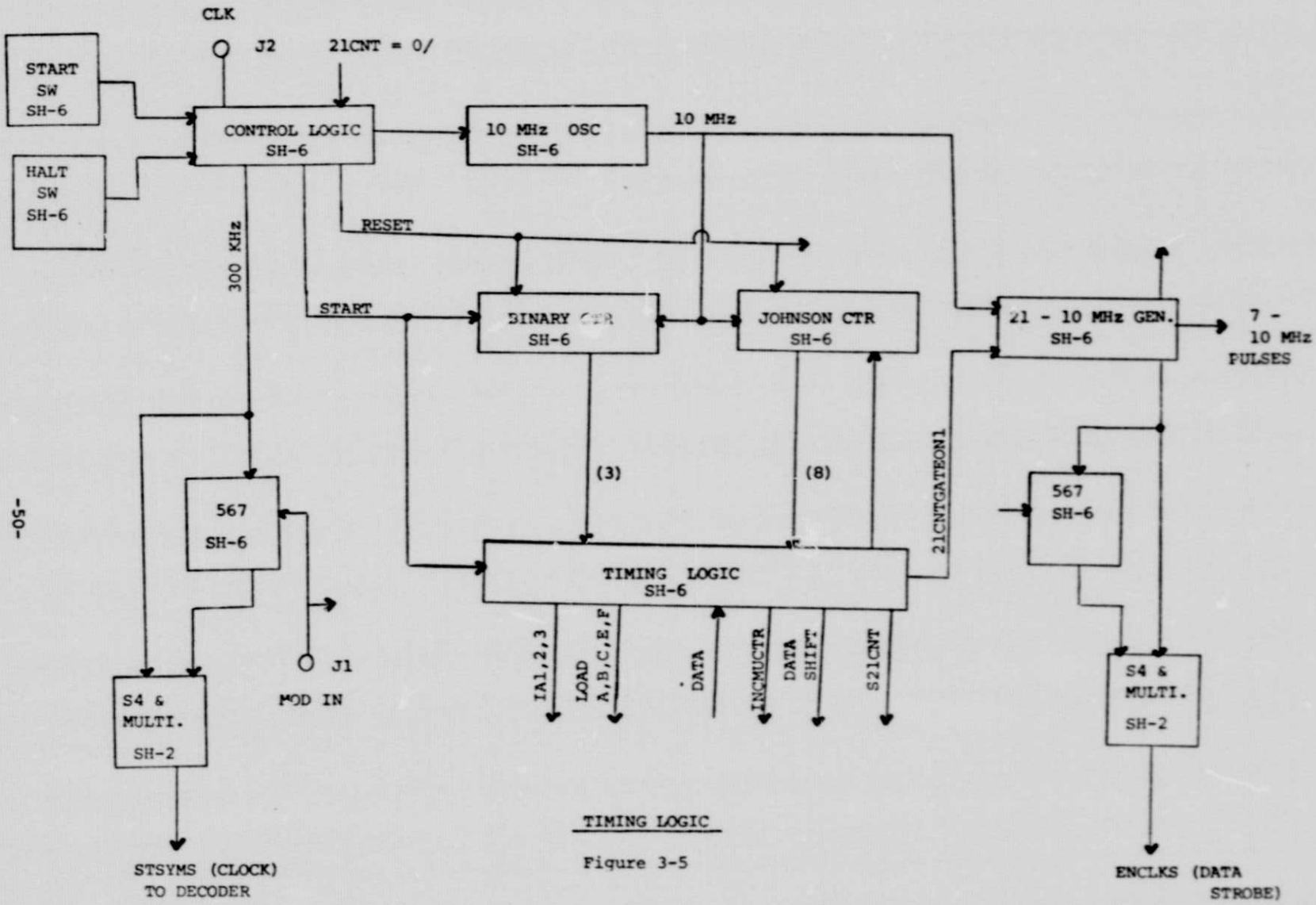
Start switch will first load the A, B, C, E and F registers with the noise data from the ROM's, next the noise generator shift registers are shifted 7 times. The resultant data from the Noise Generator Encoder (Figure 3-3) is stored on the 3 bit lines, the 300 KHz clock will sample the data at the Viterbi Decoder. The 7-burst counter is preset and a second burst of 7 pulses is applied to the noise generator SR. Upon completion of this second burst the second group of data occurs as levels on the 3 bit lines. The multiplex counter has been incremented each time to select sequentially each of the three lines to the multiplexer. The sequence is repeated a third time after which the third set of data levels appears on the three bit lines. After the third 300 KHz square wave pulse a Data Shift pulse is generated moving the data in the Random Data SR one place. Now all the aforementioned sequences repeat except the memory registers contents are not changed.

STATE OF 2-STAGE CTR	MSB'S OF MEMORY ADD.	MEMORY	DATA STORED IN
00	00	M11, M12, M13	Reg. A
01	01	M11, M12, M13	Reg. B
10	10	M11, M12, M13	Reg. C
11	11	M11, M12, M13	M11, M12, M13 (D)
00	00	M21, M22, M23	Reg. E
01	01	M21, M22, M23	Reg. F
10	10	M21, M22, M23	M21, M22, M23 (G)
11	10	M21, M22, M23	M21, M22, M23 (G)

(D) = Memory Acts as Reg. D.

(G) = Memory Acts as Reg. G.

Figure 3-4



TIMING LOGIC

Figure 3-5

IV. TEST AND TRADEOFF RESULTS

Output Selection Algorithm Tradeoffs

The error rates observed using various path lengths and output selection algorithms are plotted in Figure 4-1 as a function of the signal-to-noise ratio (i.e., ratio of signal enerty to noise spectral density). The results summarized in Figure 4-1 were obtained using a soft-decision threshold spacing of $0.6\ \sigma_n$. The effect of varying this threshold spacing is indicated in Figure 4-2. The major conclusions to be gleaned from these results are: (1) The majority vote output selection algorithm results in a performance degradation of about 0.05 dB relative to the minimum metric algorithm. (Spot checks indicate that the arbitrary oldest-bit algorithm, in contrast, results in degradation of the order of 0.3 dB). (2) The improvement obtained by increasing the path length from 32 to 39 bits in negligible using either the minimum-metric or the majority-vote algorithm. (3) The optimum soft-decision threshold spacing is between $0.5\ \sigma_n$ and $0.6\ \sigma_n$ for most signal-to-noise ratios of interest. The optimum spacing does increase slowly with signal-to-noise ration, however. On the basis of the results observed at high signal-to-noise ratios, we conjecture that a spacing of $0.7\ \sigma_n$ or greater is optimum for ratios exceeding 5 dB. In any case, the minimum-metric and the majority-vote algorithms are both relatively insensitive to the threshold spacing for any spacing in the range of $0.4\ \sigma_n$ to $0.7\ \sigma_n$.

Since the majority-vote output selection algorithm is only slightly inferior to the minimum-metric rule and since the advantage of increasing the path length from 32 to 39 bits is negligible, the majority-vote method was implemented in the brassboard and the path memory limited to 32 bits. The increase in hardware complexity associated with either the minimum-metric rule or with the increased path length does not appear to be justified by these slight improvements in performance.

Branch Synchronizer Tradeoffs

The performance of the baseline branch synchronizer is depicted in Figure 4-3. The two curves represent the synchronization delay in information bits as a function of the signal-to-noise ratio for two up-down counter thresholds, T = 32, and T = 64 (cf. Section 2). The dashed lines indicate the synchronization delay to the first acceptance; the solid lines indicate the delay to the first correct acceptance. In both cases, the synchronizer was able to retain synchronization at all test set signal-to-noise ratio settings greater than -3 dB.

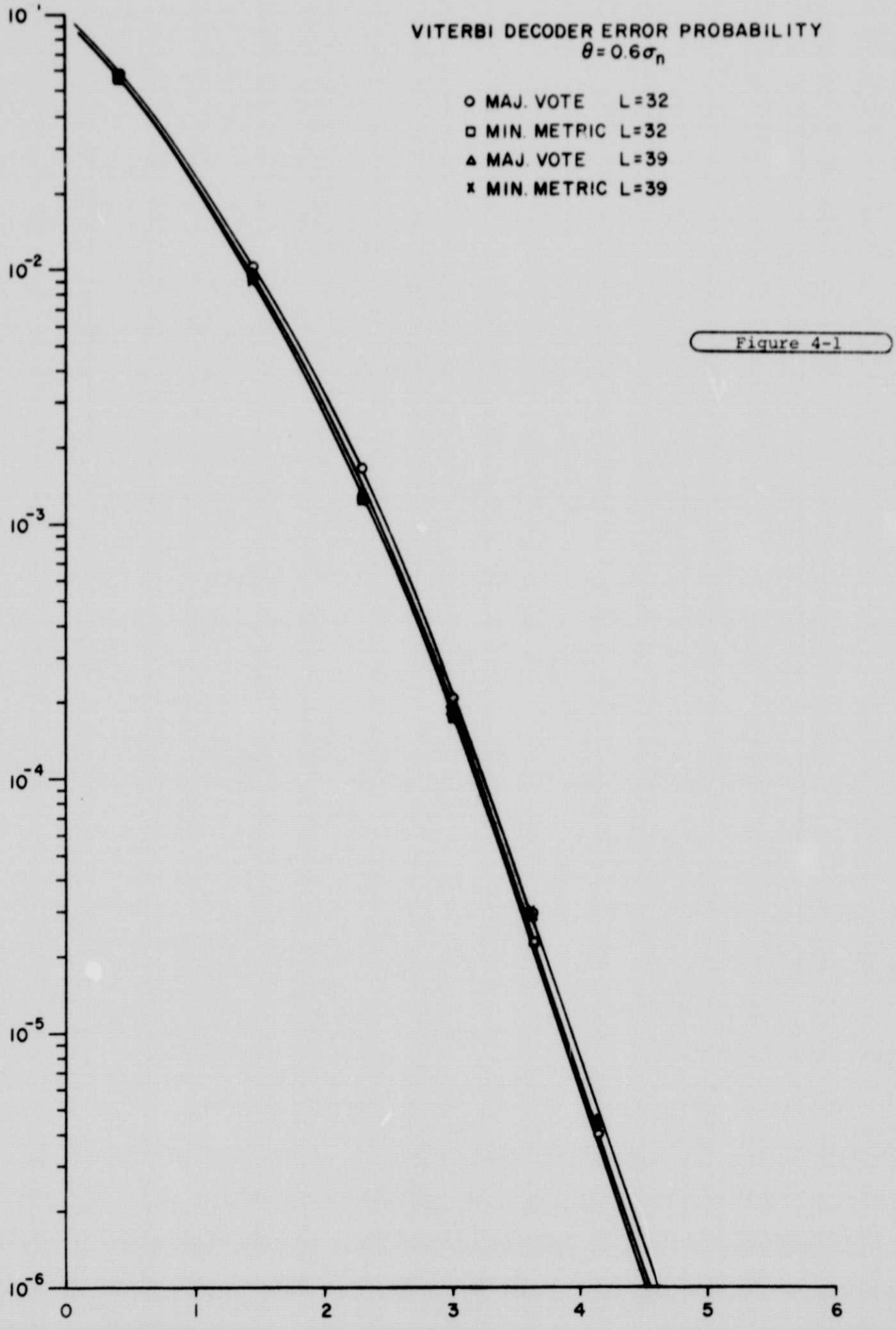

VITERBI DECODER ERROR PROBABILITY
$\theta = 0.6\sigma_n$
o MAJ. VOTE L=32
□ MIN. METRIC L=32
Δ MAJ. VOTE L=39
x MIN. METRIC L=39
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0
1
2
3
4
5
6
S/N (dB)

Figure 4-1

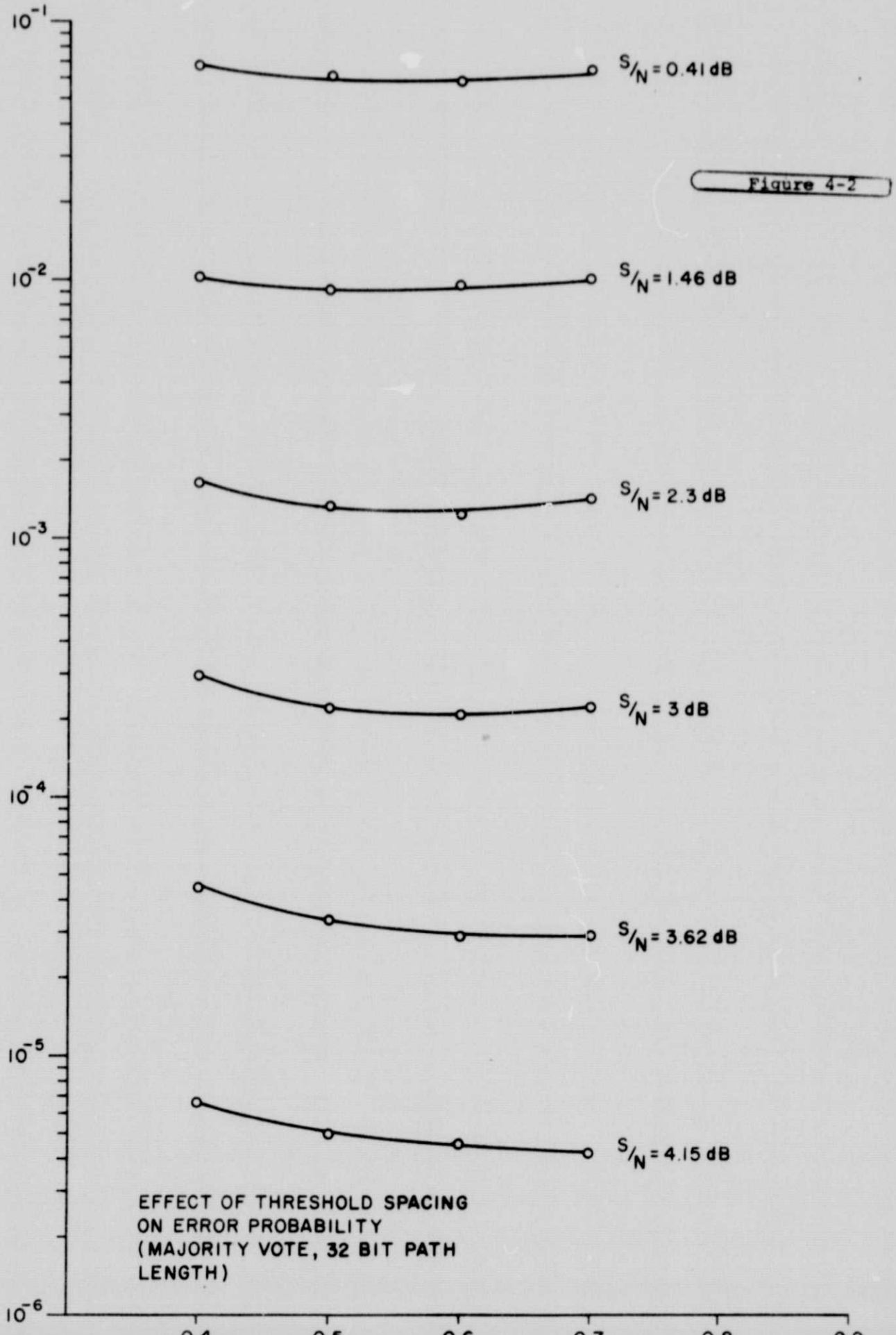


Figure 4-2

EFFECT OF THRESHOLD SPACING ON ERROR PROBABILITY (MAJORITY VOTE, 32 BIT PATH LENGTH)

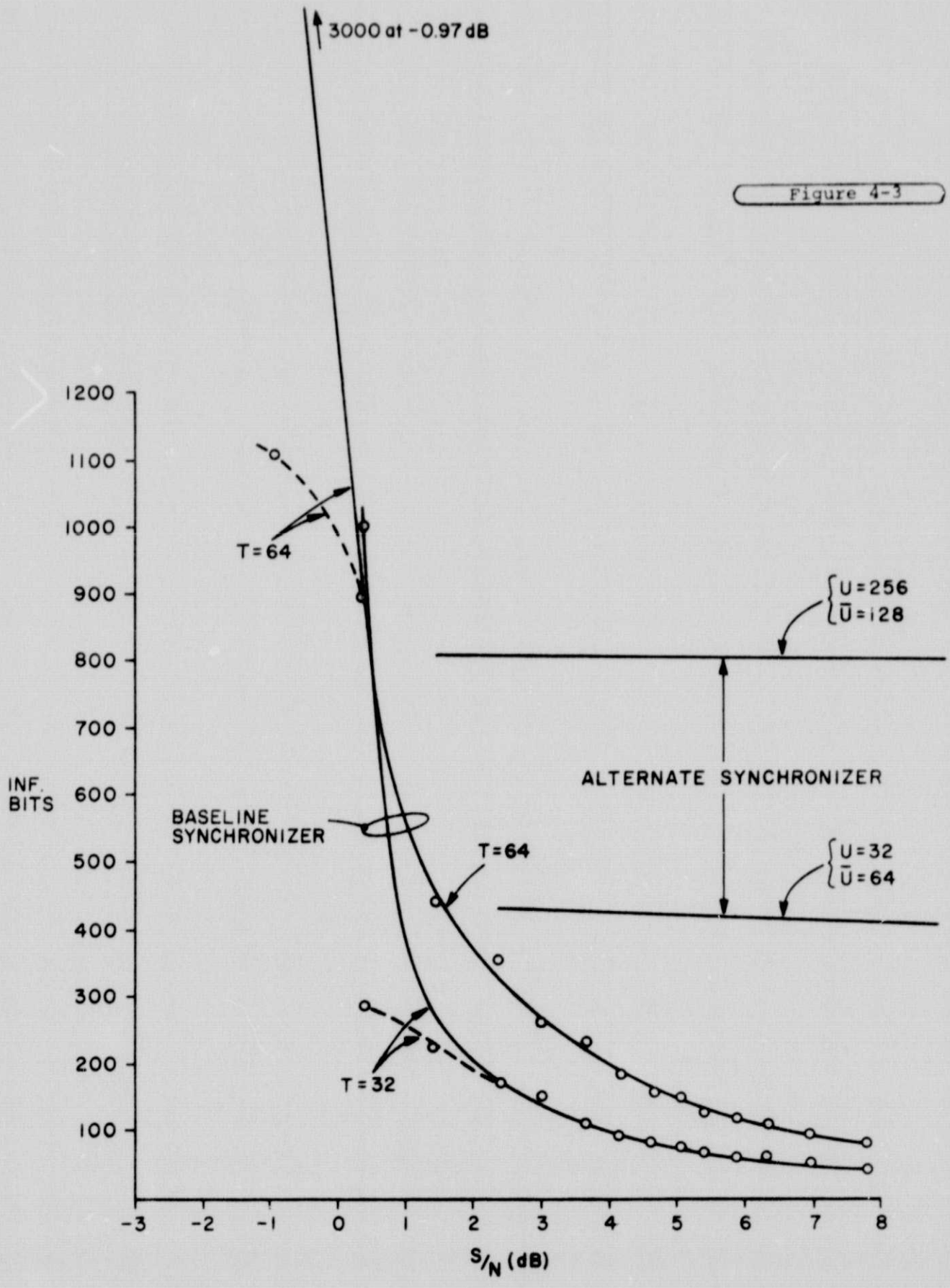

3000 at -0.97 dB
1200
1100
1000
900
800
700
600
500
400
300
200
100
INF. BITS
T=64
U=256
Ū=128
ALTERNATE SYNCHRONIZER
BASELINE SYNCHRONIZER
T=64
U=32
Ū=64
T=32
-3
-2
-1
0
1
2
3
4
5
6
7
8
S/N (dB)

Figure 4-3

A more conventional branch synchronizer was breadboarded and tested in order to establish performance data against which to compare the baseline synchronizer. This synchronizer is conceptually straightforward and not difficult to implement. It functions by allowing the Viterbi Decoder to begin decoding at an arbitrary position (i.e., at any one of the three possible branch sync positions with any one of the two possible polarities) and, after a ℓ-bit delay (with ℓ the path memory length), it monitors the oldest bits in each of the 64 paths. If all of these bits are the same when an output bit is selected, it augments a "unanimity counter"; if not, it augments a "non-unanimity counter." If the unanimity counter reaches the count of U before the non-unanimity counter reaches $\overline{U}$, it accepts the current position (and polarity) as correct. If the non-unanimity counter first reaches $\overline{U}$, either the sync position is slipped or the polarity changed and the procedure begins anew. The position and polarity are alternately changed until an acceptable (i.e., presumably the correct) position and polarity are found.

Various values of U and $\overline{U}$ were tested in order to find the most efficacious settings. Table 4-1 lists the lowest signal-to-noise ratio at which the resulting synchronizer could retain branch synchronization as a function of U and $\overline{U}$. The propensity for false-sync is also indicated in Table 4-1. This refers to the tendency for the synchronizer temporarily to accept an erroneous sync position during acquisition. In every case, however, the false position is quickly rejected and correct sync eventually found. This, of course, increases the amount of time needed for correct synchronization. Nevertheless, a small U, $\overline{U}$ setting might result in a faster sync acquisition than a larger setting, even though the former involves a number of false attempts before it finally succeeds, since each individual decision requires less time.

Acquisition delay measurements were made with U = 32 and $\overline{U}$ = 64 and with U = 256 and $\overline{U}$ = 128 in order to estimate the efficacy of this approach. Since false syncs made direct measurement of the acquisition time difficult, the following indirect method was used: Acquisition delay measurements were taken under two conditions, one in which the synchronizer was set to begin testing the correct position immediately and one in which it was set so that it had to test all six possibilities before reaching the correct position. In the latter case, the number m of positions tested and rejected before the first acceptance (either correct or false sync) was also recorded.

TABLE 4-1

Counter Settings for Alternate Branch Synchronizer

U	$\overline{U}$	Frequency of False Syncs	Lowest S/N Ratio Without Loosing Sync
32	64	High	2.3 dB
128	128	Moderate	1.46 dB
128	256	High	1.46 dB
256	128	Zero	1.46 dB
256	256	Low	1.46 dB
512	256	Zero	1.46 dB
512	512	Low	0.41 dB

Since

$$E(m) = \beta\left\{(1-\beta) + 2(1-\beta)^2 + 3(1-\beta)^3 + 4(1-\beta)^4\right\} + 5(1-\alpha)(1-\beta)^5$$

with β the probability of false acceptance and α the probability of false rejection, and since $\alpha \cong 0$ for all U and $\overline{U}$ settings of concern here, the average $\hat{m}$ of the numbers m thus provides an estimate $\hat{\beta}$ of β. Further, since the expected number of false syncs before correct sync when ℓ false positions are tested before the correct position is tested is

$$\sum_{i=1}^{\infty} \binom{\ell+i-1}{i} \beta^i \quad (1-\beta)^{\ell} = \frac{\ell\beta}{1-\beta}$$

the expected search time is

$$\hat{n}_1 + \frac{1}{6} \sum_{\ell=0}^{5} \frac{\ell\beta}{1-\beta} \hat{n}_2 + \frac{1}{6} \sum_{\ell=0}^{5} \ell \hat{n}_3$$

$$= \hat{n}_1 + \frac{5}{2} \frac{\beta}{1-\beta} \hat{n}_2 + {}^5/_2 \hat{n}_3$$

where n_1 is the expected test time (in information bits) needed to accept the correct sync position, n_2 the expected time to accept an incorrect sync position and n_3 the expected time to reject an incorrect position. The quantities n_1, n_2, and n_3 were directly estimated by, respectively, the average number of bits needed for a decision when the correct position was the first tested, the average number needed when the first incorrect position tested was accepted, and the average of the number of bits needed to make the first acceptance minus the expected number needed for that final test (n_1 or n_2) divided by the number of rejections prior to the first acceptance.

The results of these measurements and calculations showed an expected sync acquisition time remarkably independent of the signal-to-noise ratio. When U = 32, $\overline{U}$ = 64, the average acquisition delay varied from 430 bits at $^S/_N$ = 2.3 dB to 400 bits at an infinite signal-to-noise ratio. Similarly, when U = 256, $\overline{U}$ = 128, the average delay remained almost constant at 800 bits as the signal-to-noise ratio ranged from 1.46 dB to infinity. These results are also indicated on Figure 4-2.

This alternate branch synchronization technique thus appears to be reasonably effective. Since it is slower than the baseline approach, however, and since it is unable to retain synchronization at low signal-to-noise ratios, the baseline approach was implemented in the brassboard. The up-down counter threshold T was set at 32 since the resulting synchronizer is faster, at most signal-to-noise ratios of interest, than when T = 64. This setting is easily changed, however, should more rapid acquisition be desired at signal-to-noise ratios of less than 1 dB.

Power Consumption

Power consumption calculated from measured current for the decoder totaled approximately 2.6 watts, with the power breakdown per voltage as follows:

10 VDC	@ 180 ma	= 1.8	watts
8 VDC	@ 65 ma	= 0.52	watts
5 VDC	@ 60 ma	= 0.30	watts
	TOTAL	= 0.62	watts

In the preliminary design it was planned to operate the synchronizer and self-test logic at 5 VDC to save power. In the final design, however, the number of level conversion gates required for the interface between the 5 and 10 VDC turned out to be fifteen. It was decided, therefore, that little if any power savings would be realized, considering the additional logic requirements.

The preliminary power estimation of 0.5 watts for the path memory was based on operation of the memory devices (CD4036) at 5 VDC. Interconnection capacity in the brassboard, however, requires a minimum path memory voltage of approximately 7.2 VDC for proper operation. Below this voltage the rise and fall times of the memory output signals become excessive. An 8 VDC source was, therefore, provided for the path memory from a regulator circuit driven by the 10 VDC power supply.

LSI Implementation

The Viterbi Decoder brassboard was partitioned to facilitate eventual LSI implementation. The ASC and path memories, in particular, have highly regular organizations readily translatable to either monolithic or hybrid LSI. For the relatively low production volume anticipated for the Space Vehicle Viterbi Decoder, the hybrid approach merits serious consideration. It can result in packaging densities comparable to those attainable with specially developed monolithic chips

at considerably reduced development cost. For this reason, the major emphasis concerning possible LSI implementations for the Viterbi Decoder involved the hybrid approach.

The packaging technique resulting from this study is shown in Figure 4-4. The path memory entails the use of thick-film hybrids. The 128 chips comprising the two path memories are partitioned into eight identical circuit groups, each packaged as a hybrid circuit containing 16 CMOS chips. The high density of parallel connections required in each hybrid is satisfied using three, thick-film, etched-back conductive layers. The chips are attached to the substrate through epoxy die bonding. The chip-to-substrate interconnections are ultrasonically bonded aluminum wire; the substrate-to-package interconnections are TC-bonded gold ribbon. The enclosure for this circuit is a custom-tooled flatpack with a total of 116 input-output pins. The sealed package size is approximately 1.5" x 2.0" x 0.16".

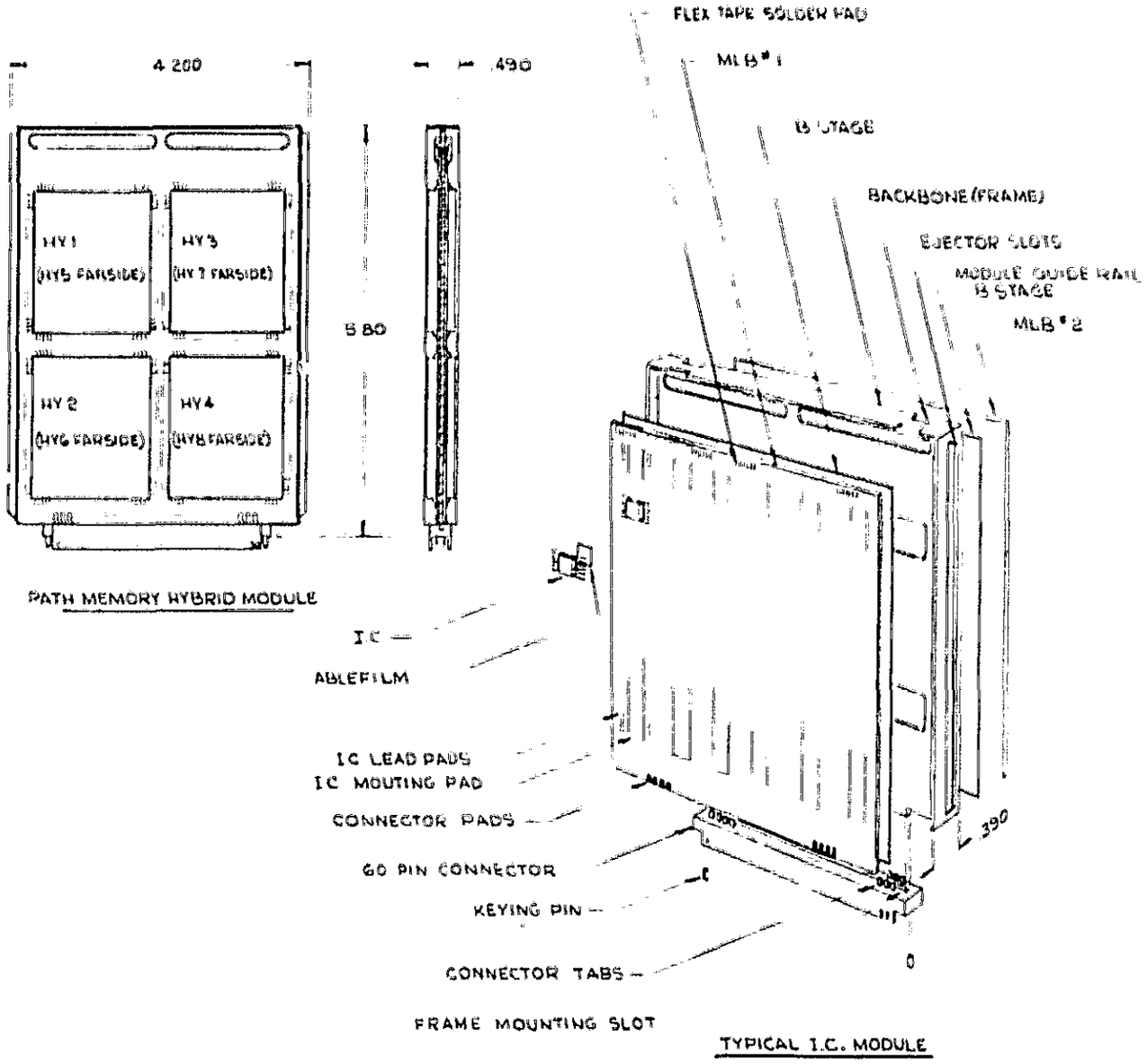
Both the ACS section and the remainder of the decoder logic (the metric calculator, node synchronizer, and control logic) could also be implemented as hybrid LSI circuits. The resulting reduction in volume is considerably less dramatic in either of these sections, however, than it is in the path memory. The ACS section and the metric-calculator-synchronizer-control section could, in fact, each be packaged on one 4.2" x 5.8" x 0.39" module using flatpack integrated circuits (cf. Figure 4-4) without recourse to any additional hybrid or monolithic LSI development. The entire hybrid path memory could then be packaged on a third module having the dimensions 4.2" x 5.8" x 0.49", resulting in an overall 38 $in.^3$ decoder package.

The total circuit capacitance in the 38 cubic inch package just described is estimated to be less than one-half the capacitance in the brassboard decoder. Since the dynamic power dissipation is directly proportional to the load capacitance, the total power would be roughly one-half the predicted attainable brassboard value of 2.6 watts, or approximately 1.3 watts.

Maximum Decoder Bit Rate

The maximum data bit rate in the brassboard decoder is approximately 99 KBPS. Logic delays in the sequencer start/stop logic prevent operation at a 100 KBPS bit rate with the included 6.5 MHz crystal oscillator. The decoder was operated at approximately 105 KBPS, however, using an external pulse generator in place of the internal oscillator. This indicates that the substitution of an oscillator

of slightly higher frequency would easily allow operation at a data rate of 100 KBPS or more.



ELECTRONIC MODULES (SRU'S)

Figure 4-4

V. CONCLUSIONS

The Space Vehicle Viterbi Decoder Brassboard has demonstrated that a low-power, rate 1/3, constraint-lenght 7 Viterbi decodor capable of operation of rates up to 100 KBPS is indeed within the current state-of-the-art. Were the same design implemented using flatpacks for the ACS and control sections of the decoder and hybrid packages for the path memory, the resulting decoder would occupy only 38 cubic inches and consume an estimated 1.3 watts of power. Furthermore, this significant reduction in volume could be achieved at a modest cost since only one hybrid package need be developed; no monolithic LSI chip developments are required.

In addition to validating the design concept, the brassboard effort also accomplished a number of other objectives. It is verified, for example, that the optimum soft-decision threshold spacing is between $0.5\ \sigma_n$ and $0.6\ \sigma_n$ for most signal-to-noise ratios of interest and that the optimum ratio of the spacing to the noise standard deviation σ_n is a slowly increasing function of the signal-to-noise ratio. It established that the difference between the majority-vote output selection rule and the minimum-metric rule is indeed small (the former performing within about 0.05 dB of the latter at all signal-to-noise ratios) and that the advantage of increasing the path length from 32 bits to 39 bits is entirely negligible. (Th. arbitrary-oldest-path output selection rule, in contrast, was found to introduce degradations of roughly 0.3 dB.)

Finally, it verified that the stand-alone node synchronizer implemented as part of the brassboard is indeed an effective device resulting, for example, in an average node acquisition delay of less than 200 bits at a 3 dB signal-to-noise ratio and capable of acquiring and retaining nodes synchronization at signal-to-noise ratios of as low as -1 dB.